# *TMS320F2837xD Dual-Core Real-Time Microcontrollers*

*Technical Reference Manual*

Literature Number: SPRUHM8K
DECEMBER 2013 – REVISED MAY 2024

# Table of Contents


**Read This First** ..... 73
About This Manual ..... 73
Notational Conventions ..... 73
Glossary ..... 73
Related Documentation From Texas Instruments ..... 73
Support Resources ..... 73
Trademarks ..... 74
**1 C2000™ Microcontrollers Software Support** ..... 75
1.1 Introduction ..... 76
1.2 C2000Ware Structure ..... 76
1.3 Documentation ..... 76
1.4 Devices ..... 76
1.5 Libraries ..... 76
1.6 Code Composer Studio™ Integrated Development Environment (IDE) ..... 76
1.7 SysConfig and PinMUX Tool ..... 77
**2 C28x Processor** ..... 78
2.1 Introduction ..... 79
2.2 C28X Related Collateral ..... 79
2.3 Features ..... 79
2.4 Floating-Point Unit ..... 80
2.5 Trigonometric Math Unit (TMU) ..... 80
2.6 Viterbi, Complex Math, and CRC Unit II (VCU-II) ..... 81
**3 System Control and Interrupt** ..... 82
3.1 Introduction ..... 83
3.2 System Control Functional Description ..... 83
3.2.1 Device Identification ..... 83
3.2.2 Device Configuration Registers ..... 84
3.3 Resets ..... 84
3.3.1 Reset Sources ..... 84
3.3.2 External Reset ($\overline{\text{XRS}}$) ..... 85
3.3.3 Power-On Reset (POR) ..... 85
3.3.4 Debugger Reset ($\overline{\text{SYSRS}}$) ..... 85
3.3.5 Watchdog Reset ($\overline{\text{WDRS}}$) ..... 86
3.3.6 NMI Watchdog Reset ($\overline{\text{NMIWDRS}}$) ..... 86
3.3.7 DCSM Safe Code Copy Reset ($\overline{\text{SCCRESET}}$) ..... 86
3.3.8 Hibernate Reset ($\overline{\text{HIBRESET}}$) ..... 86
3.3.9 Hardware BIST Reset ($\overline{\text{HWBISTRS}}$) ..... 86
3.3.10 Test Reset ($\overline{\text{TRST}}$) ..... 86
3.4 Peripheral Interrupts ..... 87
3.4.1 Interrupt Concepts ..... 87
3.4.2 Interrupt Architecture ..... 87
3.4.3 Interrupt Entry Sequence ..... 89
3.4.4 Configuring and Using Interrupts ..... 90
3.4.5 PIE Channel Mapping ..... 92
3.4.6 Vector Tables ..... 94
3.5 Exceptions and Non-Maskable Interrupts ..... 99
3.5.1 Configuring and Using NMIs ..... 99
3.5.2 Emulation Considerations ..... 100
3.5.3 NMI Sources ..... 100





Copyright © 2024 Texas Instruments Incorporated

3.5.4 Illegal Instruction Trap (ITRAP) ... 100
3.6 Safety Features ... 101
3.6.1 Write Protection on Registers ... 101
3.6.2 Missing Clock Detection Logic ... 101
3.6.3 PLLSLIP Detection ... 102
3.6.4 CPU1 and CPU2 PIE Vector Address Validity Check ... 103
3.6.5 NMIWDs ... 103
3.6.6 ECC and Parity Enabled RAMs, Shared RAMs Protection ... 103
3.6.7 ECC Enabled Flash Memory ... 103
3.6.8 ERRORSTS Pin ... 104
3.7 Clocking ... 104
3.7.1 Clock Sources ... 106
3.7.2 Derived Clocks ... 107
3.7.3 Device Clock Domains ... 108
3.7.4 XCLKOUT ... 109
3.7.5 Clock Connectivity ... 110
3.7.6 Clock Source and PLL Setup ... 111
3.7.7 Clock (OSCCLK) Failure Detection ... 114
3.8 32-Bit CPU Timers 0/1/2 ... 116
3.9 Watchdog Timers ... 118
3.9.1 Servicing the Watchdog Timer ... 119
3.9.2 Minimum Window Check ... 119
3.9.3 Watchdog Reset or Watchdog Interrupt Mode ... 120
3.9.4 Watchdog Operation in Low-Power Modes ... 120
3.9.5 Emulation Considerations ... 120
3.10 Low-Power Modes ... 121
3.10.1 IDLE ... 121
3.10.2 STANDBY ... 121
3.10.3 HALT ... 122
3.10.4 Hibernate (HIB) ... 124
3.11 Memory Controller Module ... 125
3.11.1 Functional Description ... 125
3.12 Flash and OTP Memory ... 133
3.12.1 Features ... 133
3.12.2 Flash Tools ... 133
3.12.3 Default Flash Configuration ... 134
3.12.4 Flash Bank, One-Time Programmable (OTP) Memory, and Flash Pump ... 134
3.12.5 Flash Module Controller (FMC) ... 135
3.12.6 Flash and OTP Memory Power-Down Modes and Wakeup ... 135
3.12.7 Flash and OTP Memory Performance ... 137
3.12.8 Flash Read Interface ... 137
3.12.9 Erase/Program Flash ... 139
3.12.10 Error Correction Code (ECC) Protection ... 140
3.12.11 Reserved Locations Within Flash and OTP Memory ... 144
3.12.12 Procedure to Change the Flash Control Registers ... 144
3.12.13 Simple Procedure to Modify an Application from RAM Configuration to Flash Configuration ... 144
3.12.14 Flash Pump Ownership Semaphore ... 145
3.13 Dual Code Security Module (DCSM) ... 147
3.13.1 Functional Description ... 147
3.13.2 CSM Impact on Other On-Chip Resources ... 154
3.13.3 Incorporating Code Security in User Applications ... 155
3.14 JTAG ... 160
3.15 System Control Register Configuration Restrictions ... 160
3.16 Software ... 161
3.16.1 SYSCTL Examples ... 161
3.16.2 TIMER Examples ... 161
3.16.3 MEMCFG Examples ... 162
3.16.4 INTERRUPT Examples ... 162
3.16.5 LPM Examples ... 165
3.16.6 WATCHDOG Examples ... 165
3.17 System Control Registers ... 166



Copyright © 2024 Texas Instruments Incorporated

3.17.1 System Control Base Addresses....166
3.17.2 CPUTIMER_REGS Registers....167
3.17.3 PIE_CTRL_REGS Registers....174
3.17.4 WD_REGS Registers....226
3.17.5 NMI_INTRUPT_REGS Registers....232
3.17.6 XINT_REGS Registers....245
3.17.7 SYNC_SOC_REGS Registers....254
3.17.8 DMA_CLA_SRC_SEL_REGS Registers....261
3.17.9 FLASH_PUMP_SEMAPHORE_REGS Registers....268
3.17.10 DEV_CFG_REGS Registers....270
3.17.11 CLK_CFG_REGS Registers....337
3.17.12 CPU_SYS_REGS Registers....360
3.17.13 ROM_PREFETCH_REGS Registers....399
3.17.14 DCSM_Z1_REGS Registers....401
3.17.15 DCSM_Z2_REGS Registers....421
3.17.16 DCSM_COMMON_REGS Registers....441
3.17.17 MEM_CFG_REGS Registers....448
3.17.18 ACCESS_PROTECTION_REGS Registers....495
3.17.19 MEMORY_ERROR_REGS Registers....518
3.17.20 ROM_WAIT_STATE_REGS Registers....535
3.17.21 FLASH_CTRL_REGS Registers....537
3.17.22 FLASH_ECC_REGS Registers....546
3.17.23 CPU_ID_REGS Registers....569
3.17.24 UID_REGS Registers....571
3.17.25 DCSM_Z1_OTP Registers....580
3.17.26 DCSM_Z2_OTP Registers....588
3.17.27 Register to Driverlib Function Mapping....596
**4 ROM Code and Peripheral Booting**....614
4.1 Introduction....615
4.2 Boot ROM Registers....615
4.3 Device Boot Sequence....615
4.4 Device Boot Modes....616
4.5 Configuring Boot Mode Pins....617
4.6 Configuring Get Boot Options....619
4.7 Configuring Emulation Boot Options....620
4.8 Device Boot Flow Diagrams....621
4.8.1 Emulation Boot Flow Diagrams....623
4.8.2 Standalone and Hibernate Boot Flow Diagrams....625
4.9 Device Reset and Exception Handling....627
4.9.1 Reset Causes and Handling....627
4.9.2 Exceptions and Interrupts Handling....628
4.10 Boot ROM Description....629
4.10.1 Entry Points....629
4.10.2 Wait Points....629
4.10.3 Memory Maps....630
4.10.4 Boot Modes....633
4.10.5 Boot Data Stream Structure....648
4.10.6 GPIO Assignments....650
4.10.7 Secure ROM Function APIs....652
4.10.8 Boot IPC....653
4.10.9 Clock Initializations....658
4.10.10 Wait State Configuration....659
4.10.11 Boot Status information....659
4.10.12 ROM Version....662
**5 Direct Memory Access (DMA)**....663
5.1 Introduction....664
5.1.1 Features....664
5.1.2 Block Diagram....665
5.2 Architecture....666
5.2.1 Common Peripheral Architecture....666
5.2.2 Peripheral Interrupt Event Trigger Sources....667



Copyright © 2024 Texas Instruments Incorporated

5.2.3 DMA Bus ... 671
5.3 Address Pointer and Transfer Control ... 671
5.4 Pipeline Timing and Throughput ... 677
5.5 CPU and CLA Arbitration ... 678
5.6 Channel Priority ... 679
5.6.1 Round-Robin Mode ... 679
5.6.2 Channel 1 High-Priority Mode ... 680
5.7 Overrun Detection Feature ... 680
5.8 Software ... 681
5.8.1 DMA Examples ... 681
5.9 DMA Registers ... 682
5.9.1 DMA Base Addresses ... 682
5.9.2 DMA_REGS Registers ... 683
5.9.3 DMA_CH_REGS Registers ... 688
5.9.4 DMA Registers to Driverlib Functions ... 714
**6 Control Law Accelerator (CLA)** ... 717
6.1 Introduction ... 718
6.1.1 Features ... 718
6.1.2 CLA Related Collateral ... 718
6.1.3 Block Diagram ... 719
6.2 CLA Interface ... 720
6.2.1 CLA Memory ... 720
6.2.2 CLA Memory Bus ... 721
6.2.3 Shared Peripherals and EALLOW Protection ... 722
6.2.4 CLA Tasks and Interrupt Vectors ... 722
6.2.5 CLA Software Interrupt to CPU ... 725
6.3 CLA and CPU Arbitration ... 726
6.3.1 CLA Message RAM ... 726
6.3.2 CLA Program Memory ... 727
6.3.3 CLA Data Memory ... 728
6.3.4 Peripheral Registers (ePWM, HRPWM, Comparator) ... 728
6.4 CLA Configuration and Debug ... 729
6.4.1 Building a CLA Application ... 729
6.4.2 Typical CLA Initialization Sequence ... 729
6.4.3 Debugging CLA Code ... 730
6.4.4 CLA Illegal Opcode Behavior ... 731
6.4.5 Resetting the CLA ... 732
6.5 Pipeline ... 732
6.5.1 Pipeline Overview ... 732
6.5.2 CLA Pipeline Alignment ... 733
6.5.3 Parallel Instructions ... 737
6.5.4 CLA Task Execution Latency ... 737
6.6 Software ... 738
6.6.1 CLA Examples ... 738
6.7 Instruction Set ... 739
6.7.1 Instruction Descriptions ... 739
6.7.2 Addressing Modes and Encoding ... 740
6.7.3 Instructions ... 743
6.8 CLA Registers ... 870
6.8.1 CLA Base Addresses ... 870
6.8.2 CLA_REGS Registers ... 871
6.8.3 CLA_SOFTINT_REGS Registers ... 910
6.8.4 CLA Registers to Driverlib Functions ... 914
**7 Interprocessor Communication (IPC)** ... 916
7.1 Introduction ... 917
7.2 Message RAMs ... 918
7.3 IPC Flags and Interrupts ... 918
7.4 IPC Command Registers ... 918
7.5 Free-Running Counter ... 918
7.6 IPC Communication Protocol ... 919
7.7 IPC Registers ... 920



Copyright © 2024 Texas Instruments Incorporated

7.7.1 IPC Base Addresses ... 920
7.7.2 IPC_REGS_CPU1 Registers ... 921
7.7.3 IPC_REGS_CPU2 Registers ... 953
7.7.4 IPC Registers to Driverlib Functions ... 984
**8 General-Purpose Input/Output (GPIO)** ... 986
8.1 Introduction ... 987
8.1.1 GPIO Related Collateral ... 988
8.2 Configuration Overview ... 989
8.3 Digital General-Purpose I/O Control ... 990
8.4 Input Qualification ... 991
8.4.1 No Synchronization (Asynchronous Input) ... 991
8.4.2 Synchronization to SYSCLKOUT Only ... 991
8.4.3 Qualification Using a Sampling Window ... 992
8.5 USB Signals ... 995
8.6 SPI Signals ... 995
8.7 GPIO and Peripheral Muxing ... 996
8.7.1 GPIO Muxing ... 996
8.7.2 Peripheral Muxing ... 1002
8.8 Internal Pullup Configuration Requirements ... 1004
8.9 Software ... 1005
8.9.1 GPIO Examples ... 1005
8.9.2 LED Examples ... 1005
8.10 GPIO Registers ... 1006
8.10.1 GPIO Base Addresses ... 1006
8.10.2 GPIO_CTRL_REGS Registers ... 1007
8.10.3 GPIO_DATA_REGS Registers ... 1173
8.10.4 GPIO Registers to Driverlib Functions ... 1222
**9 Crossbar (X-BAR)** ... 1229
9.1 Input X-BAR ... 1230
9.2 ePWM, CLB, and GPIO Output X-BAR ... 1233
9.2.1 ePWM X-BAR ... 1233
9.2.2 CLB X-BAR ... 1235
9.2.3 GPIO Output X-BAR ... 1238
9.2.4 X-BAR Flags ... 1240
9.3 XBAR Registers ... 1241
9.3.1 XBAR Base Addresses ... 1241
9.3.2 INPUT_XBAR_REGS Registers ... 1242
9.3.3 XBAR_REGS Registers ... 1259
9.3.4 EPWM_XBAR_REGS Registers ... 1272
9.3.5 CLB_XBAR_REGS Registers ... 1365
9.3.6 OUTPUT_XBAR_REGS Registers ... 1458
9.3.7 Register to Driverlib Function Mapping ... 1559
**10 Analog Subsystem** ... 1565
10.1 Introduction ... 1566
10.1.1 Features ... 1566
10.1.2 Block Diagram ... 1566
10.2 Optimizing Power-Up Time ... 1570
10.3 Analog Subsystem Registers ... 1571
10.3.1 Analog Subsystem Base Addresses ... 1571
10.3.2 ANALOG_SUBSYS_REGS Registers ... 1572
**11 Analog-to-Digital Converter (ADC)** ... 1582
11.1 Introduction ... 1583
11.1.1 ADC Related Collateral ... 1583
11.1.2 Features ... 1584
11.1.3 Block Diagram ... 1585
11.2 ADC Configurability ... 1586
11.2.1 Clock Configuration ... 1586
11.2.2 Resolution ... 1586
11.2.3 Voltage Reference ... 1587
11.2.4 Signal Mode ... 1587
11.2.5 Expected Conversion Results ... 1588



Copyright © 2024 Texas Instruments Incorporated

11.2.6 Interpreting Conversion Results....1589
11.3 SOC Principle of Operation....1590
11.3.1 SOC Configuration....1591
11.3.2 Trigger Operation....1591
11.3.3 ADC Acquisition (Sample and Hold) Window....1591
11.3.4 ADC Input Models....1592
11.3.5 Channel Selection....1593
11.4 SOC Configuration Examples....1594
11.4.1 Single Conversion from ePWM Trigger....1594
11.4.2 Oversampled Conversion from ePWM Trigger....1594
11.4.3 Multiple Conversions from CPU Timer Trigger....1595
11.4.4 Software Triggering of SOCs....1596
11.5 ADC Conversion Priority....1596
11.6 Burst Mode....1599
11.6.1 Burst Mode Example....1599
11.6.2 Burst Mode Priority Example....1600
11.7 EOC and Interrupt Operation....1601
11.7.1 Interrupt Overflow....1602
11.7.2 Continue to Interrupt Mode....1602
11.7.3 Early Interrupt Configuration Mode....1602
11.8 Post-Processing Blocks....1603
11.8.1 PPB Offset Correction....1604
11.8.2 PPB Error Calculation....1604
11.8.3 PPB Limit Detection and Zero-Crossing Detection....1604
11.8.4 PPB Sample Delay Capture....1606
11.9 Opens/Shorts Detection Circuit (OSDETECT)....1607
11.9.1 Implementation....1608
11.9.2 Detecting an Open Input Pin....1608
11.9.3 Detecting a Shorted Input Pin....1608
11.10 Power-Up Sequence....1609
11.11 ADC Calibration....1609
11.11.1 ADC Zero Offset Calibration....1610
11.11.2 ADC Calibration Routines in OTP Memory....1611
11.12 ADC Timings....1612
11.12.1 ADC Timing Diagrams....1612
11.13 Additional Information....1618
11.13.1 Ensuring Synchronous Operation....1618
11.13.2 Choosing an Acquisition Window Duration....1622
11.13.3 Achieving Simultaneous Sampling....1624
11.13.4 Result Register Mapping....1624
11.13.5 Internal Temperature Sensor....1624
11.13.6 Designing an External Reference Circuit....1625
11.14 Software....1627
11.14.1 ADC Examples....1627
11.15 ADC Registers....1630
11.15.1 ADC Base Addresses....1630
11.15.2 ADC_RESULT_REGS Registers....1631
11.15.3 ADC_REGS Registers....1652
11.15.4 ADC Registers to Driverlib Functions....1784
**12 Buffered Digital-to-Analog Converter (DAC)....1789**
12.1 Introduction....1790
12.1.1 DAC Related Collateral....1790
12.1.2 Features....1790
12.1.3 Block Diagram....1790
12.2 Using the DAC....1791
12.2.1 Initialization Sequence....1791
12.2.2 DAC Offset Adjustment....1792
12.2.3 EPWMSYNCPER Signal....1792
12.3 Lock Registers....1792
12.4 Software....1793
12.4.1 DAC Examples....1793



Copyright © 2024 Texas Instruments Incorporated

12.5 DAC Registers ... 1793
12.5.1 DAC Base Addresses ... 1793
12.5.2 DAC_REGS Registers ... 1794
12.5.3 DAC Registers to Driverlib Functions ... 1801
**13 Comparator Subsystem (CMPSS)** ... 1803
13.1 Introduction ... 1804
13.1.1 CMPSS Related Collateral ... 1804
13.1.2 Features ... 1804
13.1.3 Block Diagram ... 1805
13.2 Comparator ... 1805
13.3 Reference DAC ... 1806
13.4 Ramp Generator ... 1807
13.4.1 Ramp Generator Overview ... 1807
13.4.2 Ramp Generator Behavior ... 1808
13.4.3 Ramp Generator Behavior at Corner Cases ... 1809
13.5 Digital Filter ... 1810
13.5.1 Filter Initialization Sequence ... 1811
13.6 Using the CMPSS ... 1811
13.6.1 LATCHCLR and EPWMSYNCPER Signals ... 1811
13.6.2 Synchronizer, Digital Filter, and Latch Delays ... 1811
13.6.3 Calibrating the CMPSS ... 1812
13.6.4 Enabling and Disabling the CMPSS Clock ... 1812
13.7 Software ... 1813
13.7.1 CMPSS Examples ... 1813
13.8 CMPSS Registers ... 1814
13.8.1 CMPSS Base Addresses ... 1814
13.8.2 CMPSS_REGS Registers ... 1815
13.8.3 CMPSS Registers to Driverlib Functions ... 1838
**14 Sigma Delta Filter Module (SDFM)** ... 1841
14.1 Introduction ... 1842
14.1.1 SDFM Related Collateral ... 1842
14.1.2 Features ... 1843
14.1.3 Block Diagram ... 1844
14.2 Configuring Device Pins ... 1846
14.3 Input Control Unit ... 1847
14.4 Sinc Filter ... 1848
14.4.1 Data Rate and Latency of the Sinc Filter ... 1850
14.5 Data (Primary) Filter Unit ... 1851
14.5.1 32-bit or 16-bit Data Filter Output Representation ... 1852
14.5.2 SDSYNC Event ... 1853
14.6 Comparator (Secondary) Filter Unit ... 1854
14.6.1 Higher Threshold (HLT) Comparator ... 1855
14.6.2 Lower Threshold (LLT) Comparator ... 1855
14.7 Theoretical SDFM Filter Output ... 1856
14.8 Interrupt Unit ... 1858
14.8.1 SDFM (SDINT) Interrupt Sources ... 1858
14.9 Register Descriptions ... 1860
14.10 Software ... 1862
14.10.1 SDFM Examples ... 1862
14.11 SDFM Registers ... 1862
14.11.1 SDFM Base Addresses ... 1862
14.11.2 SDFM_REGS Registers ... 1863
14.11.3 SDFM Registers to Driverlib Functions ... 1899
**15 Enhanced Pulse Width Modulator (ePWM)** ... 1901
15.1 Introduction ... 1902
15.1.1 EPWM Related Collateral ... 1903
15.1.2 Submodule Overview ... 1904
15.2 Configuring Device Pins ... 1909
15.3 ePWM Modules Overview ... 1909
15.4 Time-Base (TB) Submodule ... 1911
15.4.1 Purpose of the Time-Base Submodule ... 1911



Copyright © 2024 Texas Instruments Incorporated

15.4.2 Controlling and Monitoring the Time-Base Submodule ... 1912
15.4.3 Calculating PWM Period and Frequency ... 1914
15.4.4 Phase Locking the Time-Base Clocks of Multiple ePWM Modules ... 1917
15.4.5 Simultaneous Writes to TBPRD and CMPx Registers Between ePWM Modules ... 1917
15.4.6 Time-Base Counter Modes and Timing Waveforms ... 1918
15.4.7 Global Load ... 1922
15.5 Counter-Compare (CC) Submodule ... 1924
15.5.1 Purpose of the Counter-Compare Submodule ... 1924
15.5.2 Controlling and Monitoring the Counter-Compare Submodule ... 1925
15.5.3 Operational Highlights for the Counter-Compare Submodule ... 1926
15.5.4 Count Mode Timing Waveforms ... 1927
15.6 Action-Qualifier (AQ) Submodule ... 1930
15.6.1 Purpose of the Action-Qualifier Submodule ... 1930
15.6.2 Action-Qualifier Submodule Control and Status Register Definitions ... 1931
15.6.3 Action-Qualifier Event Priority ... 1933
15.6.4 AQCTLA and AQCTLB Shadow Mode Operations ... 1934
15.6.5 Configuration Requirements for Common Waveforms ... 1936
15.7 Dead-Band Generator (DB) Submodule ... 1943
15.7.1 Purpose of the Dead-Band Submodule ... 1943
15.7.2 Dead-band Submodule Additional Operating Modes ... 1944
15.7.3 Operational Highlights for the Dead-Band Submodule ... 1946
15.8 PWM Chopper (PC) Submodule ... 1950
15.8.1 Purpose of the PWM Chopper Submodule ... 1950
15.8.2 Operational Highlights for the PWM Chopper Submodule ... 1950
15.8.3 Waveforms ... 1951
15.9 Trip-Zone (TZ) Submodule ... 1954
15.9.1 Purpose of the Trip-Zone Submodule ... 1954
15.9.2 Operational Highlights for the Trip-Zone Submodule ... 1955
15.9.3 Generating Trip Event Interrupts ... 1957
15.10 Event-Trigger (ET) Submodule ... 1960
15.10.1 Operational Overview of the ePWM Event-Trigger Submodule ... 1961
15.11 Digital Compare (DC) Submodule ... 1965
15.11.1 Purpose of the Digital Compare Submodule ... 1967
15.11.2 Enhanced Trip Action Using CMPSS ... 1967
15.11.3 Using CMPSS to Trip the ePWM on a Cycle-by-Cycle Basis ... 1967
15.11.4 Operation Highlights of the Digital Compare Submodule ... 1968
15.12 ePWM Crossbar (X-BAR) ... 1975
15.13 Applications to Power Topologies ... 1976
15.13.1 Overview of Multiple Modules ... 1976
15.13.2 Key Configuration Capabilities ... 1977
15.13.3 Controlling Multiple Buck Converters With Independent Frequencies ... 1978
15.13.4 Controlling Multiple Buck Converters With Same Frequencies ... 1980
15.13.5 Controlling Multiple Half H-Bridge (HHB) Converters ... 1982
15.13.6 Controlling Dual 3-Phase Inverters for Motors (ACI and PMSM) ... 1984
15.13.7 Practical Applications Using Phase Control Between PWM Modules ... 1986
15.13.8 Controlling a 3-Phase Interleaved DC/DC Converter ... 1987
15.13.9 Controlling Zero Voltage Switched Full Bridge (ZVSFB) Converter ... 1990
15.13.10 Controlling a Peak Current Mode Controlled Buck Module ... 1992
15.13.11 Controlling H-Bridge LLC Resonant Converter ... 1993
15.14 High-Resolution Pulse Width Modulator (HRPWM) ... 1994
15.14.1 Operational Description of HRPWM ... 1996
15.14.2 SFO Library Software - SFO_TI_Build_V8.lib ... 2017
15.15 ePWM Registers ... 2020
15.15.1 ePWM Base Addresses ... 2020
15.15.2 EPWM_REGS Registers ... 2021
15.15.3 Register to Driverlib Function Mapping ... 2141
**16 Enhanced Capture (eCAP)** ... 2153
16.1 Introduction ... 2154
16.1.1 Features ... 2154
16.1.2 ECAP Related Collateral ... 2154
16.2 Description ... 2154



Copyright © 2024 Texas Instruments Incorporated

16.3 Configuring Device Pins for the eCAP ... 2155
16.4 Capture and APWM Operating Mode ... 2156
16.5 Capture Mode Description ... 2158
16.5.1 Event Prescaler ... 2159
16.5.2 Edge Polarity Select and Qualifier ... 2159
16.5.3 Continuous/One-Shot Control ... 2160
16.5.4 32-Bit Counter and Phase Control ... 2161
16.5.5 CAP1-CAP4 Registers ... 2161
16.5.6 eCAP Synchronization ... 2161
16.5.7 Interrupt Control ... 2163
16.5.8 DMA Interrupt ... 2165
16.5.9 Shadow Load and Lockout Control ... 2165
16.5.10 APWM Mode Operation ... 2165
16.6 Application of the eCAP Module ... 2167
16.6.1 Example 1 - Absolute Time-Stamp Operation Rising-Edge Trigger ... 2167
16.6.2 Example 2 - Absolute Time-Stamp Operation Rising- and Falling-Edge Trigger ... 2168
16.6.3 Example 3 - Time Difference (Delta) Operation Rising-Edge Trigger ... 2169
16.6.4 Example 4 - Time Difference (Delta) Operation Rising- and Falling-Edge Trigger ... 2170
16.7 Application of the APWM Mode ... 2171
16.7.1 Example 1 - Simple PWM Generation (Independent Channels) ... 2171
16.8 Software ... 2172
16.8.1 ECAP Examples ... 2172
16.9 eCAP Registers ... 2173
16.9.1 eCAP Base Addresses ... 2173
16.9.2 ECAP_REGS Registers ... 2174
16.9.3 ECAP Registers to Driverlib Functions ... 2189
**17 Enhanced Quadrature Encoder Pulse (eQEP)** ... 2192
17.1 Introduction ... 2193
17.1.1 EQEP Related Collateral ... 2195
17.2 Configuring Device Pins ... 2195
17.3 Description ... 2196
17.3.1 EQEP Inputs ... 2196
17.3.2 Functional Description ... 2197
17.3.3 eQEP Memory Map ... 2198
17.4 Quadrature Decoder Unit (QDU) ... 2199
17.4.1 Position Counter Input Modes ... 2199
17.4.2 eQEP Input Polarity Selection ... 2202
17.4.3 Position-Compare Sync Output ... 2202
17.5 Position Counter and Control Unit (PCCU) ... 2202
17.5.1 Position Counter Operating Modes ... 2202
17.5.2 Position Counter Latch ... 2205
17.5.3 Position Counter Initialization ... 2207
17.5.4 eQEP Position-compare Unit ... 2208
17.6 eQEP Edge Capture Unit ... 2210
17.7 eQEP Watchdog ... 2214
17.8 eQEP Unit Timer Base ... 2214
17.9 eQEP Interrupt Structure ... 2215
17.10 eQEP Registers ... 2215
17.10.1 eQEP Base Addresses ... 2215
17.10.2 EQEP_REGS Registers ... 2216
17.10.3 EQEP Registers to Driverlib Functions ... 2247
**18 Serial Peripheral Interface (SPI)** ... 2250
18.1 Introduction ... 2251
18.1.1 Features ... 2251
18.1.2 SPI Related Collateral ... 2251
18.1.3 Block Diagram ... 2252
18.2 System-Level Integration ... 2253
18.2.1 SPI Module Signals ... 2253
18.2.2 Configuring Device Pins ... 2254
18.2.3 SPI Interrupts ... 2254
18.2.4 DMA Support ... 2256



Copyright © 2024 Texas Instruments Incorporated

18.3 SPI Operation....2257
18.3.1 Introduction to Operation....2257
18.3.2 Master Mode....2258
18.3.3 Slave Mode....2259
18.3.4 Data Format....2261
18.3.5 Baud Rate Selection....2262
18.3.6 SPI Clocking Schemes....2263
18.3.7 SPI FIFO Description....2264
18.3.8 SPI DMA Transfers....2265
18.3.9 SPI High-Speed Mode....2266
18.3.10 SPI 3-Wire Mode Description....2266
18.4 Programming Procedure....2268
18.4.1 Initialization Upon Reset....2268
18.4.2 Configuring the SPI....2268
18.4.3 Configuring the SPI for High-Speed Mode....2269
18.4.4 Data Transfer Example....2270
18.4.5 SPI 3-Wire Mode Code Examples....2271
18.4.6 SPI STEINV Bit in Digital Audio Transfers....2273
18.5 Software....2274
18.5.1 SPI Examples....2274
18.6 SPI Registers....2277
18.6.1 SPI Base Addresses....2277
18.6.2 SPI_REGS Registers....2278
18.6.3 SPI Registers to Driverlib Functions....2296
**19 Serial Communications Interface (SCI)....2298**
19.1 Introduction....2299
19.1.1 Features....2299
19.1.2 SCI Related Collateral....2300
19.1.3 Block Diagram....2300
19.2 Architecture....2300
19.3 SCI Module Signal Summary....2300
19.4 Configuring Device Pins....2302
19.5 Multiprocessor and Asynchronous Communication Modes....2302
19.6 SCI Programmable Data Format....2303
19.7 SCI Multiprocessor Communication....2304
19.7.1 Recognizing the Address Byte....2304
19.7.2 Controlling the SCI TX and RX Features....2304
19.7.3 Receipt Sequence....2304
19.8 Idle-Line Multiprocessor Mode....2305
19.8.1 Idle-Line Mode Steps....2305
19.8.2 Block Start Signal....2306
19.8.3 Wake-Up Temporary (WUT) Flag....2306
19.8.4 Receiver Operation....2306
19.9 Address-Bit Multiprocessor Mode....2307
19.9.1 Sending an Address....2307
19.10 SCI Communication Format....2308
19.10.1 Receiver Signals in Communication Modes....2309
19.10.2 Transmitter Signals in Communication Modes....2310
19.11 SCI Port Interrupts....2311
19.11.1 Break Detect....2312
19.12 SCI Baud Rate Calculations....2312
19.13 SCI Enhanced Features....2313
19.13.1 SCI FIFO Description....2313
19.13.2 SCI Auto-Baud....2315
19.13.3 Autobaud-Detect Sequence....2315
19.14 Software....2316
19.14.1 SCI Examples....2316
19.15 SCI Registers....2316
19.15.1 SCI Base Addresses....2316
19.15.2 SCI_REGS Registers....2317
19.15.3 SCI Registers to Driverlib Functions....2338



Copyright © 2024 Texas Instruments Incorporated

**20 Inter-Integrated Circuit Module (I2C)** ..... 2341
20.1 Introduction ..... 2342
20.1.1 I2C Related Collateral ..... 2342
20.1.2 Features ..... 2343
20.1.3 Features Not Supported ..... 2343
20.1.4 Functional Overview ..... 2344
20.1.5 Clock Generation ..... 2345
20.1.6 I2C Clock Divider Registers (I2CCLKL and I2CCLKH) ..... 2346
20.2 Configuring Device Pins ..... 2347
20.3 I2C Module Operational Details ..... 2347
20.3.1 Input and Output Voltage Levels ..... 2347
20.3.2 Selecting Pullup Resistors ..... 2347
20.3.3 Data Validity ..... 2347
20.3.4 Operating Modes ..... 2347
20.3.5 I2C Module START and STOP Conditions ..... 2351
20.3.6 Non-repeat Mode versus Repeat Mode ..... 2352
20.3.7 Serial Data Formats ..... 2352
20.3.8 Clock Synchronization ..... 2355
20.3.9 Arbitration ..... 2356
20.3.10 Digital Loopback Mode ..... 2357
20.3.11 NACK Bit Generation ..... 2358
20.4 Interrupt Requests Generated by the I2C Module ..... 2358
20.4.1 Basic I2C Interrupt Requests ..... 2359
20.4.2 I2C FIFO Interrupts ..... 2362
20.5 Resetting or Disabling the I2C Module ..... 2362
20.6 Software ..... 2363
20.6.1 I2C Examples ..... 2363
20.7 I2C Registers ..... 2365
20.7.1 I2C Base Addresses ..... 2365
20.7.2 I2C_REGS Registers ..... 2366
20.7.3 I2C Registers to Driverlib Functions ..... 2389
**21 Multichannel Buffered Serial Port (McBSP)** ..... 2391
21.1 Introduction ..... 2392
21.1.1 MCBSP Related Collateral ..... 2392
21.1.2 Features of the McBSPs ..... 2392
21.1.3 McBSP Pins/Signals ..... 2393
21.2 Configuring Device Pins ..... 2393
21.3 McBSP Operation ..... 2394
21.3.1 Data Transfer Process of McBSPs ..... 2395
21.3.2 Companding (Compressing and Expanding) Data ..... 2395
21.3.3 Clocking and Framing Data ..... 2397
21.3.4 Frame Phases ..... 2399
21.3.5 McBSP Reception ..... 2402
21.3.6 McBSP Transmission ..... 2403
21.3.7 Interrupts and DMA Events Generated by a McBSP ..... 2404
21.4 McBSP Sample Rate Generator ..... 2404
21.4.1 Block Diagram ..... 2405
21.4.2 Frame Synchronization Generation in the Sample Rate Generator ..... 2408
21.4.3 Synchronizing Sample Rate Generator Outputs to an External Clock ..... 2408
21.4.4 Reset and Initialization Procedure for the Sample Rate Generator ..... 2410
21.5 McBSP Exception/Error Conditions ..... 2411
21.5.1 Types of Errors ..... 2411
21.5.2 Overrun in the Receiver ..... 2412
21.5.3 Unexpected Receive Frame-Synchronization Pulse ..... 2413
21.5.4 Overwrite in the Transmitter ..... 2415
21.5.5 Underflow in the Transmitter ..... 2416
21.5.6 Unexpected Transmit Frame-Synchronization Pulse ..... 2417
21.6 Multichannel Selection Modes ..... 2420
21.6.1 Channels, Blocks, and Partitions ..... 2420
21.6.2 Multichannel Selection ..... 2421
21.6.3 Configuring a Frame for Multichannel Selection ..... 2421



Copyright © 2024 Texas Instruments Incorporated

21.6.4 Using Two Partitions....2421
21.6.5 Using Eight Partitions....2423
21.6.6 Receive Multichannel Selection Mode....2424
21.6.7 Transmit Multichannel Selection Modes....2425
21.6.8 Using Interrupts Between Block Transfers....2426
21.7 SPI Operation Using the Clock Stop Mode....2428
21.7.1 SPI Protocol....2428
21.7.2 Clock Stop Mode....2428
21.7.3 Enable and Configure the Clock Stop Mode....2429
21.7.4 Clock Stop Mode Timing Diagrams....2430
21.7.5 Procedure for Configuring a McBSP for SPI Operation....2432
21.7.6 McBSP as the SPI Master....2432
21.7.7 McBSP as an SPI Slave....2434
21.8 Receiver Configuration....2435
21.8.1 Programming the McBSP Registers for the Desired Receiver Operation....2435
21.8.2 Resetting and Enabling the Receiver....2436
21.8.3 Set the Receiver Pins to Operate as McBSP Pins....2437
21.8.4 Digital Loopback Mode....2437
21.8.5 Clock Stop Mode....2437
21.8.6 Receive Multichannel Selection Mode....2438
21.8.7 Receive Frame Phases....2438
21.8.8 Receive Word Lengths....2439
21.8.9 Receive Frame Length....2440
21.8.10 Receive Frame-Synchronization Ignore Function....2441
21.8.11 Receive Companding Mode....2442
21.8.12 Receive Data Delay....2444
21.8.13 Receive Sign-Extension and Justification Mode....2446
21.8.14 Receive Interrupt Mode....2447
21.8.15 Receive Frame-Synchronization Mode....2447
21.8.16 Receive Frame-Synchronization Polarity....2449
21.8.17 Receive Clock Mode....2452
21.8.18 Receive Clock Polarity....2453
21.8.19 SRG Clock Divide-Down Value....2455
21.8.20 SRG Clock Synchronization Mode....2455
21.8.21 SRG Clock Mode (Choose an Input Clock)....2456
21.8.22 SRG Input Clock Polarity....2456
21.9 Transmitter Configuration....2457
21.9.1 Programming the McBSP Registers for the Desired Transmitter Operation....2457
21.9.2 Resetting and Enabling the Transmitter....2458
21.9.3 Set the Transmitter Pins to Operate as McBSP Pins....2458
21.9.4 Digital Loopback Mode....2459
21.9.5 Clock Stop Mode....2459
21.9.6 Transmit Multichannel Selection Mode....2460
21.9.7 XCERs Used in the Transmit Multichannel Selection Mode....2461
21.9.8 Transmit Frame Phases....2463
21.9.9 Transmit Word Lengths....2463
21.9.10 Transmit Frame Length....2464
21.9.11 Enable/Disable the Transmit Frame-Synchronization Ignore Function....2465
21.9.12 Transmit Companding Mode....2466
21.9.13 Transmit Data Delay....2467
21.9.14 Transmit DXENA Mode....2469
21.9.15 Transmit Interrupt Mode....2469
21.9.16 Transmit Frame-Synchronization Mode....2470
21.9.17 Transmit Frame-Synchronization Polarity....2471
21.9.18 SRG Frame-Synchronization Period and Pulse Width....2472
21.9.19 Transmit Clock Mode....2473
21.9.20 Transmit Clock Polarity....2473
21.10 Emulation and Reset Considerations....2475
21.10.1 McBSP Emulation Mode....2475
21.10.2 Resetting and Initializing McBSPs....2475
21.11 Data Packing Examples....2478



Copyright © 2024 Texas Instruments Incorporated

21.11.1 Data Packing Using Frame Length and Word Length ... 2478
21.11.2 Data Packing Using Word Length and the Frame-Synchronization Ignore Function ... 2480
21.12 Interrupt Generation ... 2481
21.12.1 McBSP Receive Interrupt Generation ... 2481
21.12.2 McBSP Transmit Interrupt Generation ... 2482
21.12.3 Error Flags ... 2482
21.13 McBSP Modes ... 2483
21.14 Special Case: External Device is the Transmit Frame Master ... 2483
21.15 Software ... 2485
21.15.1 MCBSP Examples ... 2485
21.16 McBSP Registers ... 2485
21.16.1 McBSP Base Addresses ... 2485
21.16.2 McBSP_REGS Registers ... 2486
21.16.3 MCBSP Registers to Driverlib Functions ... 2530
**22 Controller Area Network (CAN)** ... 2534
22.1 Introduction ... 2535
22.1.1 DCAN Related Collateral ... 2535
22.1.2 Features ... 2535
22.1.3 Block Diagram ... 2536
22.2 Functional Description ... 2538
22.2.1 Configuring Device Pins ... 2538
22.2.2 Address/Data Bus Bridge ... 2538
22.3 Operating Modes ... 2540
22.3.1 Initialization ... 2540
22.3.2 CAN Message Transfer (Normal Operation) ... 2541
22.3.3 Test Modes ... 2542
22.4 Multiple Clock Source ... 2546
22.5 Interrupt Functionality ... 2547
22.5.1 Message Object Interrupts ... 2547
22.5.2 Status Change Interrupts ... 2547
22.5.3 Error Interrupts ... 2547
22.5.4 Peripheral Interrupt Expansion (PIE) Module Nomenclature for DCAN Interrupts ... 2547
22.5.5 Interrupt Topologies ... 2548
22.6 Parity Check Mechanism ... 2549
22.6.1 Behavior on Parity Error ... 2549
22.7 Debug Mode ... 2550
22.8 Module Initialization ... 2550
22.9 Configuration of Message Objects ... 2551
22.9.1 Configuration of a Transmit Object for Data Frames ... 2551
22.9.2 Configuration of a Transmit Object for Remote Frames ... 2551
22.9.3 Configuration of a Single Receive Object for Data Frames ... 2551
22.9.4 Configuration of a Single Receive Object for Remote Frames ... 2552
22.9.5 Configuration of a FIFO Buffer ... 2552
22.10 Message Handling ... 2552
22.10.1 Message Handler Overview ... 2553
22.10.2 Receive/Transmit Priority ... 2553
22.10.3 Transmission of Messages in Event Driven CAN Communication ... 2553
22.10.4 Updating a Transmit Object ... 2554
22.10.5 Changing a Transmit Object ... 2554
22.10.6 Acceptance Filtering of Received Messages ... 2555
22.10.7 Reception of Data Frames ... 2555
22.10.8 Reception of Remote Frames ... 2555
22.10.9 Reading Received Messages ... 2555
22.10.10 Requesting New Data for a Receive Object ... 2556
22.10.11 Storing Received Messages in FIFO Buffers ... 2556
22.10.12 Reading from a FIFO Buffer ... 2556
22.11 CAN Bit Timing ... 2558
22.11.1 Bit Time and Bit Rate ... 2558
22.11.2 Configuration of the CAN Bit Timing ... 2563
22.12 Message Interface Register Sets ... 2567
22.12.1 Message Interface Register Sets 1 and 2 (IF1 and IF2) ... 2567



Copyright © 2024 Texas Instruments Incorporated

22.12.2 Message Interface Register Set 3 (IF3)....2568
22.13 Message RAM....2569
22.13.1 Structure of Message Objects....2569
22.13.2 Addressing Message Objects in RAM....2572
22.13.3 Message RAM Representation in Debug Mode....2573
22.14 Software....2574
22.14.1 CAN Examples....2574
22.15 CAN Registers....2574
22.15.1 CAN Base Addresses....2574
22.15.2 CAN_REGS Registers....2575
22.15.3 CAN Registers to Driverlib Functions....2631
**23 Universal Serial Bus (USB) Controller**....2635
23.1 Introduction....2636
23.1.1 Features....2636
23.1.2 USB Related Collateral....2636
23.1.3 Block Diagram....2637
23.2 Functional Description....2639
23.2.1 Operation as a Device....2639
23.2.2 Operation as a Host....2644
23.2.3 DMA Operation....2648
23.2.4 Address/Data Bus Bridge....2648
23.3 Initialization and Configuration....2650
23.3.1 Pin Configuration....2650
23.3.2 Endpoint Configuration....2651
23.4 USB Global Interrupts....2651
23.5 Software....2652
23.5.1 USB Examples....2652
23.6 USB Registers....2652
23.6.1 USB Base Address....2652
23.6.2 USB Register Map....2652
23.6.3 Register Descriptions....2660
23.6.4 USB Registers to Driverlib Functions....2731
**24 Universal Parallel Port (uPP)**....2748
24.1 Introduction....2749
24.1.1 Features Supported....2749
24.2 Configuring Device Pins....2750
24.3 Functional Description....2750
24.3.1 Functional Block Diagram....2750
24.3.2 Data Flow....2751
24.3.3 Clock Generation and Control....2752
24.4 IO Interface and System Requirements....2753
24.4.1 Pin Multiplexing....2753
24.4.2 Internal DMA Controller Description....2753
24.4.3 Protocol Description....2755
24.4.4 Data Format....2758
24.4.5 Reset Considerations....2759
24.4.6 Interrupt Support....2759
24.4.7 Emulation Considerations....2760
24.4.8 Transmit and Receive FIFOs....2760
24.4.9 Transmit and Receive Data (MSG) RAM....2761
24.4.10 Initialization and Operation....2762
24.5 UPP Registers....2763
24.5.1 UPP Base Addresses....2763
24.5.2 UPP_REGS Registers....2764
24.5.3 UPP Registers to Driverlib Functions....2798
**25 External Memory Interface (EMIF)**....2801
25.1 Introduction....2802
25.1.1 Purpose of the Peripheral....2803
25.1.2 EMIF Related Collateral....2803
25.1.3 Features....2804
25.1.4 Functional Block Diagram....2805



Copyright © 2024 Texas Instruments Incorporated

25.1.5 Configuring Device Pins ... 2805
25.2 EMIF Module Architecture ... 2806
25.2.1 EMIF Clock Control ... 2806
25.2.2 EMIF Requests ... 2806
25.2.3 EMIF Signal Descriptions ... 2807
25.2.4 EMIF Signal Multiplexing Control ... 2808
25.2.5 SDRAM Controller and Interface ... 2808
25.2.6 Asynchronous Controller and Interface ... 2821
25.2.7 Data Bus Parking ... 2833
25.2.8 Reset and Initialization Considerations ... 2833
25.2.9 Interrupt Support ... 2833
25.2.10 DMA Event Support ... 2834
25.2.11 EMIF Signal Multiplexing ... 2834
25.2.12 Memory Map ... 2834
25.2.13 Priority and Arbitration ... 2835
25.2.14 System Considerations ... 2835
25.2.15 Power Management ... 2836
25.2.16 Emulation Considerations ... 2836
25.3 Example Configuration ... 2836
25.3.1 Hardware Interface ... 2836
25.3.2 Software Configuration ... 2837
25.4 EMIF Registers ... 2844
25.4.1 EMIF Base Addresses ... 2844
25.4.2 EMIF_REGS Registers ... 2845
25.4.3 EMIF1_CONFIG_REGS Registers ... 2865
25.4.4 EMIF2_CONFIG_REGS Registers ... 2870
25.4.5 EMIF Registers to Driverlib Functions ... 2873
**26 Configurable Logic Block (CLB)** ... 2875
26.1 Introduction ... 2876
26.1.1 CLB Related Collateral ... 2876
26.2 Description ... 2876
26.2.1 CLB Clock ... 2878
26.3 CLB Input/Output Connection ... 2879
26.3.1 Overview ... 2879
26.3.2 CLB Input Selection ... 2879
26.3.3 CLB Output Selection ... 2885
26.3.4 CLB Output Signal Multiplexer ... 2886
26.4 CLB Tile ... 2887
26.4.1 Static Switch Block ... 2888
26.4.2 Counter Block ... 2890
26.4.3 FSM Block ... 2893
26.4.4 LUT4 Block ... 2895
26.4.5 Output LUT Block ... 2895
26.4.6 High Level Controller (HLC) ... 2896
26.5 CPU Interface ... 2900
26.5.1 Register Description ... 2900
26.5.2 Non-Memory Mapped Registers ... 2901
26.6 DMA Access ... 2901
26.7 Software ... 2902
26.7.1 CLB Examples ... 2902
26.8 CLB Registers ... 2905
26.8.1 CLB Base Addresses ... 2905
26.8.2 CLB_LOGIC_CONFIG_REGS Registers ... 2906
26.8.3 CLB_LOGIC_CONTROL_REGS Registers ... 2938
26.8.4 CLB_DATA_EXCHANGE_REGS Registers ... 2965
26.8.5 CLB Registers to Driverlib Functions ... 2967
**27 Revision History** ... 2970

## List of Figures

Figure 3-1. Device Interrupt Architecture ... 88



Copyright © 2024 Texas Instruments Incorporated

Figure 3-2. Interrupt Propagation Path ..... 89
Figure 3-3. Missing Clock Detection Logic ..... 102
Figure 3-4. ERRORSTS Pin Diagram ..... 104
Figure 3-5. Clocking System ..... 105
Figure 3-6. Single-ended 3.3V External Clock ..... 106
Figure 3-7. External Crystal ..... 106
Figure 3-8. External Resonator ..... 107
Figure 3-9. AUXCLKIN ..... 107
Figure 3-10. Missing Clock Detection Logic ..... 115
Figure 3-11. CPU-Timers ..... 116
Figure 3-12. CPU-Timer Interrupts Signals and Output Signal ..... 117
Figure 3-13. CPU Watchdog Timer Module ..... 118
Figure 3-14. Memory Architecture ..... 125
Figure 3-15. Arbitration Scheme on Global Shared Memories ..... 127
Figure 3-16. Arbitration Scheme on Local Shared Memories ..... 128
Figure 3-17. FMC Interface with Core, Bank, and Pump ..... 135
Figure 3-18. Flash Prefetch Mode ..... 138
Figure 3-19. ECC Logic Inputs and Outputs ..... 141
Figure 3-20. Flash Pump Semaphore (PUMPREQUEST) States and State Transitions ..... 145
Figure 3-21. Clock Configuration Semaphore (CLKSEM) State Transitions ..... 146
Figure 3-22. Storage of Zone-Select Bits in OTP Memory ..... 151
Figure 3-23. Location of Zone-Select Block Based on Link-Pointer ..... 152
Figure 3-24. CSM Password Match Flow (PMF) ..... 156
Figure 3-25. ECSL Password Match Flow (PMF) ..... 158
Figure 3-26. TIM Register ..... 168
Figure 3-27. PRD Register ..... 169
Figure 3-28. TCR Register ..... 170
Figure 3-29. TPR Register ..... 172
Figure 3-30. TPRH Register ..... 173
Figure 3-31. PIECTRL Register ..... 176
Figure 3-32. PIEACK Register ..... 177
Figure 3-33. PIEIER1 Register ..... 178
Figure 3-34. PIEIFR1 Register ..... 180
Figure 3-35. PIEIER2 Register ..... 182
Figure 3-36. PIEIFR2 Register ..... 184
Figure 3-37. PIEIER3 Register ..... 186
Figure 3-38. PIEIFR3 Register ..... 188
Figure 3-39. PIEIER4 Register ..... 190
Figure 3-40. PIEIFR4 Register ..... 192
Figure 3-41. PIEIER5 Register ..... 194
Figure 3-42. PIEIFR5 Register ..... 196
Figure 3-43. PIEIER6 Register ..... 198
Figure 3-44. PIEIFR6 Register ..... 200
Figure 3-45. PIEIER7 Register ..... 202
Figure 3-46. PIEIFR7 Register ..... 204
Figure 3-47. PIEIER8 Register ..... 206
Figure 3-48. PIEIFR8 Register ..... 208
Figure 3-49. PIEIER9 Register ..... 210
Figure 3-50. PIEIFR9 Register ..... 212
Figure 3-51. PIEIER10 Register ..... 214
Figure 3-52. PIEIFR10 Register ..... 216
Figure 3-53. PIEIER11 Register ..... 218
Figure 3-54. PIEIFR11 Register ..... 220
Figure 3-55. PIEIER12 Register ..... 222
Figure 3-56. PIEIFR12 Register ..... 224
Figure 3-57. SCSR Register ..... 227
Figure 3-58. WDCNTR Register ..... 228
Figure 3-59. WDKEY Register ..... 229
Figure 3-60. WDCR Register ..... 230
Figure 3-61. WDWCR Register ..... 231
Figure 3-62. NMICFG Register ..... 233



Copyright © 2024 Texas Instruments Incorporated

Figure 3-63. NMIFLG Register ... 234
Figure 3-64. NMIFLGCLR Register ... 236
Figure 3-65. NMIFLGFRC Register ... 239
Figure 3-66. NMIWDCNT Register ... 241
Figure 3-67. NMIWDPRD Register ... 242
Figure 3-68. NMISHDFLG Register ... 243
Figure 3-69. XINT1CR Register ... 246
Figure 3-70. XINT2CR Register ... 247
Figure 3-71. XINT3CR Register ... 248
Figure 3-72. XINT4CR Register ... 249
Figure 3-73. XINT5CR Register ... 250
Figure 3-74. XINT1CTR Register ... 251
Figure 3-75. XINT2CTR Register ... 252
Figure 3-76. XINT3CTR Register ... 253
Figure 3-77. SYNCSELECT Register ... 255
Figure 3-78. ADCSOCOUTSELECT Register ... 257
Figure 3-79. SYNCSOCLOCK Register ... 260
Figure 3-80. CLA1TASKSRCSELLOCK Register ... 262
Figure 3-81. DMACHSRCSELLOCK Register ... 263
Figure 3-82. CLA1TASKSRCSEL1 Register ... 264
Figure 3-83. CLA1TASKSRCSEL2 Register ... 265
Figure 3-84. DMACHSRCSEL1 Register ... 266
Figure 3-85. DMACHSRCSEL2 Register ... 267
Figure 3-86. PUMPREQUEST Register ... 269
Figure 3-87. DEVCFGLOCK1 Register ... 272
Figure 3-88. PARTIDL Register ... 274
Figure 3-89. PARTIDH Register ... 276
Figure 3-90. REVID Register ... 277
Figure 3-91. DC0 Register ... 278
Figure 3-92. DC1 Register ... 279
Figure 3-93. DC2 Register ... 280
Figure 3-94. DC3 Register ... 281
Figure 3-95. DC4 Register ... 283
Figure 3-96. DC5 Register ... 284
Figure 3-97. DC6 Register ... 285
Figure 3-98. DC7 Register ... 286
Figure 3-99. DC8 Register ... 287
Figure 3-100. DC9 Register ... 288
Figure 3-101. DC10 Register ... 289
Figure 3-102. DC11 Register ... 290
Figure 3-103. DC12 Register ... 291
Figure 3-104. DC13 Register ... 292
Figure 3-105. DC14 Register ... 293
Figure 3-106. DC15 Register ... 294
Figure 3-107. DC17 Register ... 296
Figure 3-108. DC18 Register ... 297
Figure 3-109. DC19 Register ... 298
Figure 3-110. DC20 Register ... 299
Figure 3-111. PERCNF1 Register ... 301
Figure 3-112. FUSEERR Register ... 302
Figure 3-113. SOFTPRES0 Register ... 303
Figure 3-114. SOFTPRES1 Register ... 304
Figure 3-115. SOFTPRES2 Register ... 305
Figure 3-116. SOFTPRES3 Register ... 307
Figure 3-117. SOFTPRES4 Register ... 308
Figure 3-118. SOFTPRES6 Register ... 309
Figure 3-119. SOFTPRES7 Register ... 310
Figure 3-120. SOFTPRES8 Register ... 311
Figure 3-121. SOFTPRES9 Register ... 312
Figure 3-122. SOFTPRES11 Register ... 313
Figure 3-123. SOFTPRES13 Register ... 314



Copyright © 2024 Texas Instruments Incorporated

Figure 3-124. SOFTPRES14 Register....315
Figure 3-125. SOFTPRES16 Register....316
Figure 3-126. CPUSEL0 Register....317
Figure 3-127. CPUSEL1 Register....319
Figure 3-128. CPUSEL2 Register....320
Figure 3-129. CPUSEL3 Register....321
Figure 3-130. CPUSEL4 Register....322
Figure 3-131. CPUSEL5 Register....323
Figure 3-132. CPUSEL6 Register....324
Figure 3-133. CPUSEL7 Register....325
Figure 3-134. CPUSEL8 Register....326
Figure 3-135. CPUSEL9 Register....327
Figure 3-136. CPUSEL11 Register....328
Figure 3-137. CPUSEL12 Register....330
Figure 3-138. CPUSEL14 Register....332
Figure 3-139. CPU2RESCTL Register....333
Figure 3-140. RSTSTAT Register....334
Figure 3-141. LPMSTAT Register....335
Figure 3-142. SYSDBGCTL Register....336
Figure 3-143. CLKSEM Register....339
Figure 3-144. CLKCFGLOCK1 Register....340
Figure 3-145. CLKSRCCTL1 Register....342
Figure 3-146. CLKSRCCTL2 Register....344
Figure 3-147. CLKSRCCTL3 Register....346
Figure 3-148. SYSPLLCTL1 Register....347
Figure 3-149. SYSPLLMULT Register....348
Figure 3-150. SYSPLLSTS Register....349
Figure 3-151. AUXPLLCTL1 Register....350
Figure 3-152. AUXPLLMULT Register....351
Figure 3-153. AUXPLLSTS Register....352
Figure 3-154. SYSCLKDIVSEL Register....353
Figure 3-155. AUXCLKDIVSEL Register....354
Figure 3-156. PERCLKDIVSEL Register....355
Figure 3-157. XCLKOUTDIVSEL Register....356
Figure 3-158. LOSPCP Register....357
Figure 3-159. MCDCR Register....358
Figure 3-160. X1CNT Register....359
Figure 3-161. CPUSYSLOCK1 Register....362
Figure 3-162. HIBBOOTMODE Register....365
Figure 3-163. IORESTOREADDR Register....366
Figure 3-164. PIEVERRADDR Register....367
Figure 3-165. PCLKCR0 Register....368
Figure 3-166. PCLKCR1 Register....370
Figure 3-167. PCLKCR2 Register....371
Figure 3-168. PCLKCR3 Register....373
Figure 3-169. PCLKCR4 Register....374
Figure 3-170. PCLKCR6 Register....375
Figure 3-171. PCLKCR7 Register....376
Figure 3-172. PCLKCR8 Register....377
Figure 3-173. PCLKCR9 Register....378
Figure 3-174. PCLKCR10 Register....379
Figure 3-175. PCLKCR11 Register....380
Figure 3-176. PCLKCR12 Register....381
Figure 3-177. PCLKCR13 Register....382
Figure 3-178. PCLKCR14 Register....383
Figure 3-179. PCLKCR16 Register....385
Figure 3-180. SECMSEL Register....386
Figure 3-181. LPMCR Register....387
Figure 3-182. GPIOLPMSEL0 Register....389
Figure 3-183. GPIOLPMSEL1 Register....392
Figure 3-184. TMR2CLKCTL Register....395



Copyright © 2024 Texas Instruments Incorporated

Figure 3-185. RESC Register....397
Figure 3-186. ROMPREFETCH Register....400
Figure 3-187. Z1_LINKPOINTER Register....402
Figure 3-188. Z1_OTPSECLOCK Register....403
Figure 3-189. Z1_BOOTCTRL Register....404
Figure 3-190. Z1_LINKPOINTERERR Register....405
Figure 3-191. Z1_CSMKEY0 Register....406
Figure 3-192. Z1_CSMKEY1 Register....407
Figure 3-193. Z1_CSMKEY2 Register....408
Figure 3-194. Z1_CSMKEY3 Register....409
Figure 3-195. Z1_CR Register....410
Figure 3-196. Z1_GRABSECTR Register....411
Figure 3-197. Z1_GRABRAMR Register....414
Figure 3-198. Z1_EXEONLYSECTR Register....416
Figure 3-199. Z1_EXEONLYRAMR Register....419
Figure 3-200. Z2_LINKPOINTER Register....422
Figure 3-201. Z2_OTPSECLOCK Register....423
Figure 3-202. Z2_BOOTCTRL Register....424
Figure 3-203. Z2_LINKPOINTERERR Register....425
Figure 3-204. Z2_CSMKEY0 Register....426
Figure 3-205. Z2_CSMKEY1 Register....427
Figure 3-206. Z2_CSMKEY2 Register....428
Figure 3-207. Z2_CSMKEY3 Register....429
Figure 3-208. Z2_CR Register....430
Figure 3-209. Z2_GRABSECTR Register....431
Figure 3-210. Z2_GRABRAMR Register....434
Figure 3-211. Z2_EXEONLYSECTR Register....436
Figure 3-212. Z2_EXEONLYRAMR Register....439
Figure 3-213. FLSEM Register....442
Figure 3-214. SECTSTAT Register....443
Figure 3-215. RAMSTAT Register....446
Figure 3-216. DxLOCK Register....450
Figure 3-217. DxCOMMIT Register....451
Figure 3-218. DxACCPROT0 Register....452
Figure 3-219. DxTEST Register....453
Figure 3-220. DxINIT Register....454
Figure 3-221. DxINITDONE Register....455
Figure 3-222. LSxLOCK Register....456
Figure 3-223. LSxCOMMIT Register....458
Figure 3-224. LSxMSEL Register....460
Figure 3-225. LSxCLAPGM Register....462
Figure 3-226. LSxACCPROT0 Register....463
Figure 3-227. LSxACCPROT1 Register....465
Figure 3-228. LSxTEST Register....466
Figure 3-229. LSxINIT Register....468
Figure 3-230. LSxINITDONE Register....469
Figure 3-231. GSxLOCK Register....470
Figure 3-232. GSxCOMMIT Register....472
Figure 3-233. GSxMSEL Register....475
Figure 3-234. GSxACCPROT0 Register....477
Figure 3-235. GSxACCPROT1 Register....479
Figure 3-236. GSxACCPROT2 Register....481
Figure 3-237. GSxACCPROT3 Register....483
Figure 3-238. GSxTEST Register....485
Figure 3-239. GSxINIT Register....488
Figure 3-240. GSxINITDONE Register....490
Figure 3-241. MSGxTEST Register....492
Figure 3-242. MSGxINIT Register....493
Figure 3-243. MSGxINITDONE Register....494
Figure 3-244. NMAVFLG Register....497
Figure 3-245. NMAVSET Register....499



Copyright © 2024 Texas Instruments Incorporated

Figure 3-246. NMAVCLR Register....501
Figure 3-247. NMAVINTEN Register....503
Figure 3-248. NMCPURDAVADDR Register....504
Figure 3-249. NMCPUWRAVADDR Register....505
Figure 3-250. NMCPUFAVADDR Register....506
Figure 3-251. NMDMAWRAVADDR Register....507
Figure 3-252. NMCLA1RDAVADDR Register....508
Figure 3-253. NMCLA1WRAVADDR Register....509
Figure 3-254. NMCLA1FAVADDR Register....510
Figure 3-255. MAVFLG Register....511
Figure 3-256. MAVSET Register....512
Figure 3-257. MAVCLR Register....513
Figure 3-258. MAVINTEN Register....514
Figure 3-259. MCPUFAVADDR Register....515
Figure 3-260. MCPUWRAVADDR Register....516
Figure 3-261. MDMAWRAVADDR Register....517
Figure 3-262. UCERRFLG Register....519
Figure 3-263. UCERRSET Register....520
Figure 3-264. UCERRCLR Register....521
Figure 3-265. UCCPUREADDR Register....522
Figure 3-266. UCDMAREADDR Register....523
Figure 3-267. UCCLA1READDR Register....524
Figure 3-268. CERRFLG Register....525
Figure 3-269. CERRSET Register....526
Figure 3-270. CERRCLR Register....527
Figure 3-271. CCPUREADDR Register....528
Figure 3-272. CERRCNT Register....529
Figure 3-273. CERRTHRES Register....530
Figure 3-274. CEINTFLG Register....531
Figure 3-275. CEINTCLR Register....532
Figure 3-276. CEINTSET Register....533
Figure 3-277. CEINTEN Register....534
Figure 3-278. ROMWAITSTATE Register....536
Figure 3-279. FRDCNTL Register....538
Figure 3-280. FBAC Register....539
Figure 3-281. FBFALLBACK Register....540
Figure 3-282. FBPRDY Register....541
Figure 3-283. FPAC1 Register....542
Figure 3-284. FMSTAT Register....543
Figure 3-285. FRD_INTF_CTRL Register....545
Figure 3-286. ECC_ENABLE Register....548
Figure 3-287. SINGLE_ERR_ADDR_LOW Register....549
Figure 3-288. SINGLE_ERR_ADDR_HIGH Register....550
Figure 3-289. UNC_ERR_ADDR_LOW Register....551
Figure 3-290. UNC_ERR_ADDR_HIGH Register....552
Figure 3-291. ERR_STATUS Register....553
Figure 3-292. ERR_POS Register....555
Figure 3-293. ERR_STATUS_CLR Register....556
Figure 3-294. ERR_CNT Register....557
Figure 3-295. ERR_THRESHOLD Register....558
Figure 3-296. ERR_INTFLG Register....559
Figure 3-297. ERR_INTCLR Register....560
Figure 3-298. FDATAH_TEST Register....561
Figure 3-299. FDATAL_TEST Register....562
Figure 3-300. FADDR_TEST Register....563
Figure 3-301. FECC_TEST Register....564
Figure 3-302. FECC_CTRL Register....565
Figure 3-303. FOUTH_TEST Register....566
Figure 3-304. FOUTL_TEST Register....567
Figure 3-305. FECC_STATUS Register....568
Figure 3-306. CPUID Register....570



Copyright © 2024 Texas Instruments Incorporated

Figure 3-307. UID_PSRAND0 Register....572
Figure 3-308. UID_PSRAND1 Register....573
Figure 3-309. UID_PSRAND2 Register....574
Figure 3-310. UID_PSRAND3 Register....575
Figure 3-311. UID_PSRAND4 Register....576
Figure 3-312. UID_PSRAND5 Register....577
Figure 3-313. UID_UNIQUE Register....578
Figure 3-314. UID_CHECKSUM Register....579
Figure 3-315. Z1OTP_LINKPOINTER1 Register....581
Figure 3-316. Z1OTP_LINKPOINTER2 Register....582
Figure 3-317. Z1OTP_LINKPOINTER3 Register....583
Figure 3-318. Z1OTP_PSWDLOCK Register....584
Figure 3-319. Z1OTP_CRCLOCK Register....585
Figure 3-320. Z1OTP_JTAGLOCK Register....586
Figure 3-321. Z1OTP_BOOTCTRL Register....587
Figure 3-322. Z2OTP_LINKPOINTER1 Register....589
Figure 3-323. Z2OTP_LINKPOINTER2 Register....590
Figure 3-324. Z2OTP_LINKPOINTER3 Register....591
Figure 3-325. Z2OTP_PSWDLOCK Register....592
Figure 3-326. Z2OTP_CRCLOCK Register....593
Figure 3-327. Z2OTP_JTAGLOCK Register....594
Figure 3-328. Z2OTP_BOOTCTRL Register....595
Figure 4-1. Z1 and Z2 BOOTCTRL Selection....618
Figure 4-2. CPU1 Device Boot Flow....621
Figure 4-3. CPU2 Device Boot Flow....622
Figure 4-4. CPU1 Emulation Boot Flow....623
Figure 4-5. CPU2 Emulation Boot Flow....624
Figure 4-6. CPU1 Standalone and Hibernate Boot Flow....625
Figure 4-7. CPU2 Standalone and Hibernate Boot Flow....626
Figure 4-8. Overview of SCI Bootloader Operation....634
Figure 4-9. Overview of SCI Boot Function....635
Figure 4-10. SPI Loader....636
Figure 4-11. Data Transfer From EEPROM Flow....637
Figure 4-12. EEPROM Device at Address 0x50....638
Figure 4-13. Overview of I2C Boot Function....639
Figure 4-14. Random Read....640
Figure 4-15. Sequential Read....640
Figure 4-16. Overview of Parallel GPIO Bootloader Operation....641
Figure 4-17. Parallel GPIO Bootloader Handshake Protocol....642
Figure 4-18. Parallel GPIO Mode Overview....642
Figure 4-19. Parallel GPIO Mode - Host Transfer Flow....643
Figure 4-20. 8-Bit Parallel GetWord Function....644
Figure 4-21. Overview of CAN-A Bootloader Operation....645
Figure 4-22. USB Boot Flow....647
Figure 5-1. DMA Block Diagram....665
Figure 5-2. Common Peripheral Architecture....666
Figure 5-3. DMA Trigger Architecture....668
Figure 5-4. Peripheral Interrupt Trigger Input Diagram....669
Figure 5-5. DMA State Diagram....676
Figure 5-6. 3-Stage Pipeline DMA Transfer....677
Figure 5-7. 3-stage Pipeline with One Read Stall....677
Figure 5-8. Overrun Detection Logic....680
Figure 5-9. DMACTRL Register....684
Figure 5-10. DEBUGCTRL Register....685
Figure 5-11. PRIORITYCTRL1 Register....686
Figure 5-12. PRIORITYSTAT Register....687
Figure 5-13. MODE Register....689
Figure 5-14. CONTROL Register....691
Figure 5-15. BURST_SIZE Register....693
Figure 5-16. BURST_COUNT Register....694
Figure 5-17. SRC_BURST_STEP Register....695



Copyright © 2024 Texas Instruments Incorporated

Figure 5-18. DST_BURST_STEP Register ... 696
Figure 5-19. TRANSFER_SIZE Register ... 697
Figure 5-20. TRANSFER_COUNT Register ... 698
Figure 5-21. SRC_TRANSFER_STEP Register ... 699
Figure 5-22. DST_TRANSFER_STEP Register ... 700
Figure 5-23. SRC_WRAP_SIZE Register ... 701
Figure 5-24. SRC_WRAP_COUNT Register ... 702
Figure 5-25. SRC_WRAP_STEP Register ... 703
Figure 5-26. DST_WRAP_SIZE Register ... 704
Figure 5-27. DST_WRAP_COUNT Register ... 705
Figure 5-28. DST_WRAP_STEP Register ... 706
Figure 5-29. SRC_BEG_ADDR_SHADOW Register ... 707
Figure 5-30. SRC_ADDR_SHADOW Register ... 708
Figure 5-31. SRC_BEG_ADDR_ACTIVE Register ... 709
Figure 5-32. SRC_ADDR_ACTIVE Register ... 710
Figure 5-33. DST_BEG_ADDR_SHADOW Register ... 711
Figure 5-34. DST_ADDR_SHADOW Register ... 712
Figure 5-35. DST_BEG_ADDR_ACTIVE Register ... 713
Figure 5-36. DST_ADDR_ACTIVE Register ... 714
Figure 6-1. CLA Block Diagram ... 719
Figure 6-2. MVECT1 Register ... 873
Figure 6-3. MVECT2 Register ... 874
Figure 6-4. MVECT3 Register ... 875
Figure 6-5. MVECT4 Register ... 876
Figure 6-6. MVECT5 Register ... 877
Figure 6-7. MVECT6 Register ... 878
Figure 6-8. MVECT7 Register ... 879
Figure 6-9. MVECT8 Register ... 880
Figure 6-10. MCTL Register ... 881
Figure 6-11. MIFR Register ... 882
Figure 6-12. MIOVF Register ... 886
Figure 6-13. MIFRC Register ... 889
Figure 6-14. MICLR Register ... 891
Figure 6-15. MICLROVF Register ... 893
Figure 6-16. MIER Register ... 895
Figure 6-17. MIRUN Register ... 898
Figure 6-18. _MPC Register ... 900
Figure 6-19. _MAR0 Register ... 901
Figure 6-20. _MAR1 Register ... 902
Figure 6-21. _MSTF Register ... 903
Figure 6-22. _MR0 Register ... 906
Figure 6-23. _MR1 Register ... 907
Figure 6-24. _MR2 Register ... 908
Figure 6-25. _MR3 Register ... 909
Figure 6-26. SOFTINTEN Register ... 911
Figure 6-27. SOFTINTFRC Register ... 913
Figure 7-1. IPC Module Architecture ... 917
Figure 7-2. IPCACK Register ... 922
Figure 7-3. IPCSTS Register ... 925
Figure 7-4. IPCSET Register ... 929
Figure 7-5. IPCCLR Register ... 932
Figure 7-6. IPCFLG Register ... 937
Figure 7-7. IPCCOUNTERL Register ... 941
Figure 7-8. IPCCOUNTERH Register ... 942
Figure 7-9. IPCSENDCOM Register ... 943
Figure 7-10. IPCSENDADDR Register ... 944
Figure 7-11. IPCSENDDATA Register ... 945
Figure 7-12. IPCREMOTEREPLY Register ... 946
Figure 7-13. IPCRECVCOM Register ... 947
Figure 7-14. IPCRECVADDR Register ... 948
Figure 7-15. IPCRECVDATA Register ... 949



Copyright © 2024 Texas Instruments Incorporated

Figure 7-16. IPCLOCALREPLY Register....950
Figure 7-17. IPCBOOTSTS Register....951
Figure 7-18. IPCBOOTMODE Register....952
Figure 7-19. IPCACK Register....954
Figure 7-20. IPCSTS Register....957
Figure 7-21. IPCSET Register....961
Figure 7-22. IPCCLR Register....964
Figure 7-23. IPCFLG Register....969
Figure 7-24. IPCCOUNTERL Register....973
Figure 7-25. IPCCOUNTERH Register....974
Figure 7-26. IPCRECVCOM Register....975
Figure 7-27. IPCRECVADDR Register....976
Figure 7-28. IPCRECVDATA Register....977
Figure 7-29. IPCLOCALREPLY Register....978
Figure 7-30. IPCSENDCOM Register....979
Figure 7-31. IPCSENDADDR Register....980
Figure 7-32. IPCSENDDATA Register....981
Figure 7-33. IPCREMOTEREPLY Register....982
Figure 7-34. IPCBOOTSTS Register....983
Figure 7-35. IPCBOOTMODE Register....984
Figure 8-1. GPIO Logic for a Single Pin....988
Figure 8-2. Input Qualification Using a Sampling Window....992
Figure 8-3. Input Qualifier Clock Cycles....994
Figure 8-4. GPACTRL Register....1011
Figure 8-5. GPAQSEL1 Register....1012
Figure 8-6. GPAQSEL2 Register....1014
Figure 8-7. GPAMUX1 Register....1016
Figure 8-8. GPAMUX2 Register....1018
Figure 8-9. GPADIR Register....1020
Figure 8-10. GPAPUD Register....1022
Figure 8-11. GPAINV Register....1024
Figure 8-12. GPAODR Register....1026
Figure 8-13. GPAGMUX1 Register....1028
Figure 8-14. GPAGMUX2 Register....1029
Figure 8-15. GPACSEL1 Register....1030
Figure 8-16. GPACSEL2 Register....1031
Figure 8-17. GPACSEL3 Register....1032
Figure 8-18. GPACSEL4 Register....1033
Figure 8-19. GPALOCK Register....1034
Figure 8-20. GPACR Register....1036
Figure 8-21. GPBCTRL Register....1038
Figure 8-22. GPBQSEL1 Register....1039
Figure 8-23. GPBQSEL2 Register....1041
Figure 8-24. GPBMUX1 Register....1043
Figure 8-25. GPBMUX2 Register....1045
Figure 8-26. GPBDIR Register....1047
Figure 8-27. GPBPUD Register....1049
Figure 8-28. GPBINV Register....1051
Figure 8-29. GPBODR Register....1053
Figure 8-30. GPBAMSEL Register....1055
Figure 8-31. GPBGMUX1 Register....1057
Figure 8-32. GPBGMUX2 Register....1058
Figure 8-33. GPBCSEL1 Register....1059
Figure 8-34. GPBCSEL2 Register....1060
Figure 8-35. GPBCSEL3 Register....1061
Figure 8-36. GPBCSEL4 Register....1062
Figure 8-37. GPBLOCK Register....1063
Figure 8-38. GPBCR Register....1065
Figure 8-39. GPCCTRL Register....1067
Figure 8-40. GPCQSEL1 Register....1068
Figure 8-41. GPCQSEL2 Register....1070



Copyright © 2024 Texas Instruments Incorporated

Figure 8-42. GPCMUX1 Register ... 1072
Figure 8-43. GPCMUX2 Register ... 1074
Figure 8-44. GPCDIR Register ... 1076
Figure 8-45. GPCPUD Register ... 1078
Figure 8-46. GPCINV Register ... 1080
Figure 8-47. GPCODR Register ... 1082
Figure 8-48. GPCGMUX1 Register ... 1084
Figure 8-49. GPCGMUX2 Register ... 1085
Figure 8-50. GPCCSEL1 Register ... 1086
Figure 8-51. GPCCSEL2 Register ... 1087
Figure 8-52. GPCCSEL3 Register ... 1088
Figure 8-53. GPCCSEL4 Register ... 1089
Figure 8-54. GPCLOCK Register ... 1090
Figure 8-55. GPCCR Register ... 1092
Figure 8-56. GPDCTRL Register ... 1094
Figure 8-57. GPDQSEL1 Register ... 1095
Figure 8-58. GPDQSEL2 Register ... 1097
Figure 8-59. GPDMUX1 Register ... 1099
Figure 8-60. GPDMUX2 Register ... 1101
Figure 8-61. GPDDIR Register ... 1103
Figure 8-62. GPDPUD Register ... 1105
Figure 8-63. GPDINV Register ... 1107
Figure 8-64. GPDODR Register ... 1109
Figure 8-65. GPDGMUX1 Register ... 1111
Figure 8-66. GPDGMUX2 Register ... 1113
Figure 8-67. GPDCSEL1 Register ... 1115
Figure 8-68. GPDCSEL2 Register ... 1116
Figure 8-69. GPDCSEL3 Register ... 1117
Figure 8-70. GPDCSEL4 Register ... 1118
Figure 8-71. GPDLOCK Register ... 1119
Figure 8-72. GPDCR Register ... 1121
Figure 8-73. GPECTRL Register ... 1123
Figure 8-74. GPEQSEL1 Register ... 1124
Figure 8-75. GPEQSEL2 Register ... 1126
Figure 8-76. GPEMUX1 Register ... 1128
Figure 8-77. GPEMUX2 Register ... 1130
Figure 8-78. GPEDIR Register ... 1132
Figure 8-79. GPEPUD Register ... 1134
Figure 8-80. GPEINV Register ... 1136
Figure 8-81. GPEODR Register ... 1138
Figure 8-82. GPEGMUX1 Register ... 1140
Figure 8-83. GPEGMUX2 Register ... 1142
Figure 8-84. GPECSEL1 Register ... 1144
Figure 8-85. GPECSEL2 Register ... 1145
Figure 8-86. GPECSEL3 Register ... 1146
Figure 8-87. GPECSEL4 Register ... 1147
Figure 8-88. GPELOCK Register ... 1148
Figure 8-89. GPECR Register ... 1150
Figure 8-90. GPFCTRL Register ... 1152
Figure 8-91. GPFQSEL1 Register ... 1153
Figure 8-92. GPFMUX1 Register ... 1155
Figure 8-93. GPFDIR Register ... 1157
Figure 8-94. GPFPUD Register ... 1159
Figure 8-95. GPFINV Register ... 1161
Figure 8-96. GPFODR Register ... 1163
Figure 8-97. GPFGMUX1 Register ... 1165
Figure 8-98. GPFCSEL1 Register ... 1167
Figure 8-99. GPFCSEL2 Register ... 1168
Figure 8-100. GPFLOCK Register ... 1169
Figure 8-101. GPFCR Register ... 1171
Figure 8-102. GPADAT Register ... 1175



Copyright © 2024 Texas Instruments Incorporated

Figure 8-103. GPASET Register....1177
Figure 8-104. GPACLEAR Register....1179
Figure 8-105. GPATOGGLE Register....1181
Figure 8-106. GPBDAT Register....1183
Figure 8-107. GPBSET Register....1185
Figure 8-108. GPBCLEAR Register....1187
Figure 8-109. GPBTOGGLE Register....1189
Figure 8-110. GPCDAT Register....1191
Figure 8-111. GPCSET Register....1193
Figure 8-112. GPCCLEAR Register....1195
Figure 8-113. GPCTOGGLE Register....1197
Figure 8-114. GPDDAT Register....1199
Figure 8-115. GPDSET Register....1201
Figure 8-116. GPDCLEAR Register....1203
Figure 8-117. GPDTOGGLE Register....1205
Figure 8-118. GPEDAT Register....1207
Figure 8-119. GPESET Register....1209
Figure 8-120. GPECLEAR Register....1211
Figure 8-121. GPETOGGLE Register....1213
Figure 8-122. GPFDAT Register....1215
Figure 8-123. GPFSET Register....1217
Figure 8-124. GPFCLEAR Register....1219
Figure 8-125. GPFTOGGLE Register....1221
Figure 9-1. Input X-BAR....1231
Figure 9-2. ePWM X-BAR Architecture - Single Output....1233
Figure 9-3. CLB X-BAR Architecture - Single Output....1235
Figure 9-4. GPIO to CLB Tile Connections....1236
Figure 9-5. GPIO Output X-BAR Architecture....1238
Figure 9-6. X-BAR Input Sources....1240
Figure 9-7. INPUT1SELECT Register....1243
Figure 9-8. INPUT2SELECT Register....1244
Figure 9-9. INPUT3SELECT Register....1245
Figure 9-10. INPUT4SELECT Register....1246
Figure 9-11. INPUT5SELECT Register....1247
Figure 9-12. INPUT6SELECT Register....1248
Figure 9-13. INPUT7SELECT Register....1249
Figure 9-14. INPUT8SELECT Register....1250
Figure 9-15. INPUT9SELECT Register....1251
Figure 9-16. INPUT10SELECT Register....1252
Figure 9-17. INPUT11SELECT Register....1253
Figure 9-18. INPUT12SELECT Register....1254
Figure 9-19. INPUT13SELECT Register....1255
Figure 9-20. INPUT14SELECT Register....1256
Figure 9-21. INPUTSELECTLOCK Register....1257
Figure 9-22. XBARFLG1 Register....1260
Figure 9-23. XBARFLG2 Register....1262
Figure 9-24. XBARFLG3 Register....1264
Figure 9-25. XBARCLR1 Register....1266
Figure 9-26. XBARCLR2 Register....1268
Figure 9-27. XBARCLR3 Register....1270
Figure 9-28. TRIP4MUX0TO15CFG Register....1274
Figure 9-29. TRIP4MUX16TO31CFG Register....1277
Figure 9-30. TRIP5MUX0TO15CFG Register....1280
Figure 9-31. TRIP5MUX16TO31CFG Register....1283
Figure 9-32. TRIP7MUX0TO15CFG Register....1286
Figure 9-33. TRIP7MUX16TO31CFG Register....1289
Figure 9-34. TRIP8MUX0TO15CFG Register....1292
Figure 9-35. TRIP8MUX16TO31CFG Register....1295
Figure 9-36. TRIP9MUX0TO15CFG Register....1298
Figure 9-37. TRIP9MUX16TO31CFG Register....1301
Figure 9-38. TRIP10MUX0TO15CFG Register....1304



Copyright © 2024 Texas Instruments Incorporated

Figure 9-39. TRIP10MUX16TO31CFG Register ..... 1307
Figure 9-40. TRIP11MUX0TO15CFG Register ..... 1310
Figure 9-41. TRIP11MUX16TO31CFG Register ..... 1313
Figure 9-42. TRIP12MUX0TO15CFG Register ..... 1316
Figure 9-43. TRIP12MUX16TO31CFG Register ..... 1319
Figure 9-44. TRIP4MUXENABLE Register ..... 1322
Figure 9-45. TRIP5MUXENABLE Register ..... 1327
Figure 9-46. TRIP7MUXENABLE Register ..... 1332
Figure 9-47. TRIP8MUXENABLE Register ..... 1337
Figure 9-48. TRIP9MUXENABLE Register ..... 1342
Figure 9-49. TRIP10MUXENABLE Register ..... 1347
Figure 9-50. TRIP11MUXENABLE Register ..... 1352
Figure 9-51. TRIP12MUXENABLE Register ..... 1357
Figure 9-52. TRIPOUTINV Register ..... 1362
Figure 9-53. TRIPLOCK Register ..... 1364
Figure 9-54. AUXSIG0MUX0TO15CFG Register ..... 1367
Figure 9-55. AUXSIG0MUX16TO31CFG Register ..... 1370
Figure 9-56. AUXSIG1MUX0TO15CFG Register ..... 1373
Figure 9-57. AUXSIG1MUX16TO31CFG Register ..... 1376
Figure 9-58. AUXSIG2MUX0TO15CFG Register ..... 1379
Figure 9-59. AUXSIG2MUX16TO31CFG Register ..... 1382
Figure 9-60. AUXSIG3MUX0TO15CFG Register ..... 1385
Figure 9-61. AUXSIG3MUX16TO31CFG Register ..... 1388
Figure 9-62. AUXSIG4MUX0TO15CFG Register ..... 1391
Figure 9-63. AUXSIG4MUX16TO31CFG Register ..... 1394
Figure 9-64. AUXSIG5MUX0TO15CFG Register ..... 1397
Figure 9-65. AUXSIG5MUX16TO31CFG Register ..... 1400
Figure 9-66. AUXSIG6MUX0TO15CFG Register ..... 1403
Figure 9-67. AUXSIG6MUX16TO31CFG Register ..... 1406
Figure 9-68. AUXSIG7MUX0TO15CFG Register ..... 1409
Figure 9-69. AUXSIG7MUX16TO31CFG Register ..... 1412
Figure 9-70. AUXSIG0MUXENABLE Register ..... 1415
Figure 9-71. AUXSIG1MUXENABLE Register ..... 1420
Figure 9-72. AUXSIG2MUXENABLE Register ..... 1425
Figure 9-73. AUXSIG3MUXENABLE Register ..... 1430
Figure 9-74. AUXSIG4MUXENABLE Register ..... 1435
Figure 9-75. AUXSIG5MUXENABLE Register ..... 1440
Figure 9-76. AUXSIG6MUXENABLE Register ..... 1445
Figure 9-77. AUXSIG7MUXENABLE Register ..... 1450
Figure 9-78. AUXSIGOUTINV Register ..... 1455
Figure 9-79. AUXSIGLOCK Register ..... 1457
Figure 9-80. OUTPUT1MUX0TO15CFG Register ..... 1460
Figure 9-81. OUTPUT1MUX16TO31CFG Register ..... 1463
Figure 9-82. OUTPUT2MUX0TO15CFG Register ..... 1466
Figure 9-83. OUTPUT2MUX16TO31CFG Register ..... 1469
Figure 9-84. OUTPUT3MUX0TO15CFG Register ..... 1472
Figure 9-85. OUTPUT3MUX16TO31CFG Register ..... 1475
Figure 9-86. OUTPUT4MUX0TO15CFG Register ..... 1478
Figure 9-87. OUTPUT4MUX16TO31CFG Register ..... 1481
Figure 9-88. OUTPUT5MUX0TO15CFG Register ..... 1484
Figure 9-89. OUTPUT5MUX16TO31CFG Register ..... 1487
Figure 9-90. OUTPUT6MUX0TO15CFG Register ..... 1490
Figure 9-91. OUTPUT6MUX16TO31CFG Register ..... 1493
Figure 9-92. OUTPUT7MUX0TO15CFG Register ..... 1496
Figure 9-93. OUTPUT7MUX16TO31CFG Register ..... 1499
Figure 9-94. OUTPUT8MUX0TO15CFG Register ..... 1502
Figure 9-95. OUTPUT8MUX16TO31CFG Register ..... 1505
Figure 9-96. OUTPUT1MUXENABLE Register ..... 1508
Figure 9-97. OUTPUT2MUXENABLE Register ..... 1513
Figure 9-98. OUTPUT3MUXENABLE Register ..... 1518
Figure 9-99. OUTPUT4MUXENABLE Register ..... 1523



Copyright © 2024 Texas Instruments Incorporated

Figure 9-100. OUTPUT5MUXENABLE Register....1528
Figure 9-101. OUTPUT6MUXENABLE Register....1533
Figure 9-102. OUTPUT7MUXENABLE Register....1538
Figure 9-103. OUTPUT8MUXENABLE Register....1543
Figure 9-104. OUTPUTLATCH Register....1548
Figure 9-105. OUTPUTLATCHCLR Register....1550
Figure 9-106. OUTPUTLATCHFRC Register....1552
Figure 9-107. OUTPUTLATCHENABLE Register....1554
Figure 9-108. OUTPUTINV Register....1556
Figure 9-109. OUTPUTLOCK Register....1558
Figure 10-1. Analog Subsystem Block Diagram (337-Ball ZWT)....1567
Figure 10-2. Analog Subsystem Block Diagram (176-Pin PTP)....1568
Figure 10-3. Analog Subsystem Block Diagram (100-Pin PZP)....1569
Figure 10-4. INTOSC1TRIM Register....1573
Figure 10-5. INTOSC2TRIM Register....1574
Figure 10-6. TSNSCTL Register....1575
Figure 10-7. LOCK Register....1576
Figure 10-8. ANAREFTRIMA Register....1578
Figure 10-9. ANAREFTRIMB Register....1579
Figure 10-10. ANAREFTRIMC Register....1580
Figure 10-11. ANAREFTRIMD Register....1581
Figure 11-1. ADC Module Block Diagram....1585
Figure 11-2. SOC Block Diagram....1590
Figure 11-3. Single-Ended Input Model....1592
Figure 11-4. Differential Input Model....1592
Figure 11-5. Round Robin Priority Example....1597
Figure 11-6. High Priority Example....1598
Figure 11-7. Burst Priority Example....1600
Figure 11-8. ADC EOC Interrupts....1601
Figure 11-9. ADC PPB Block Diagram....1603
Figure 11-10. ADC PPB Interrupt Event....1605
Figure 11-11. Opens/Shorts Detection Circuit....1607
Figure 11-12. Input Circuit Equivalent with OSDETECT Enabled....1608
Figure 11-13. ADC Timings for 12-bit Mode in Early Interrupt Mode....1613
Figure 11-14. ADC Timings for 12-bit Mode in Late Interrupt Mode....1614
Figure 11-15. ADC Timings for 16-bit Mode in Early Interrupt Mode....1615
Figure 11-16. ADC Timings for 16-bit Mode in Late Interrupt Mode (SYSCLK Cycles)....1616
Figure 11-17. Example: Basic Synchronous Operation....1618
Figure 11-18. Example: Synchronous Operation with Multiple Trigger Sources....1619
Figure 11-19. Example: Synchronous Operation with Uneven SOC Numbers....1620
Figure 11-20. Example: Asynchronous Operation with Uneven SOC Numbers – Trigger Overflow....1620
Figure 11-21. Example: Asynchronous Operation with Different Resolutions....1621
Figure 11-22. Example: Synchronous Operation with Different Resolutions....1621
Figure 11-23. Example: Synchronous Equivalent Operation with Non-Overlapping Conversions....1622
Figure 11-24. ADC Reference System....1625
Figure 11-25. ADC Shared Reference System....1626
Figure 11-26. ADCRESULT0 Register....1632
Figure 11-27. ADCRESULT1 Register....1633
Figure 11-28. ADCRESULT2 Register....1634
Figure 11-29. ADCRESULT3 Register....1635
Figure 11-30. ADCRESULT4 Register....1636
Figure 11-31. ADCRESULT5 Register....1637
Figure 11-32. ADCRESULT6 Register....1638
Figure 11-33. ADCRESULT7 Register....1639
Figure 11-34. ADCRESULT8 Register....1640
Figure 11-35. ADCRESULT9 Register....1641
Figure 11-36. ADCRESULT10 Register....1642
Figure 11-37. ADCRESULT11 Register....1643
Figure 11-38. ADCRESULT12 Register....1644
Figure 11-39. ADCRESULT13 Register....1645
Figure 11-40. ADCRESULT14 Register....1646



Copyright © 2024 Texas Instruments Incorporated

Figure 11-41. ADCRESULT15 Register....1647
Figure 11-42. ADCPPB1RESULT Register....1648
Figure 11-43. ADCPPB2RESULT Register....1649
Figure 11-44. ADCPPB3RESULT Register....1650
Figure 11-45. ADCPPB4RESULT Register....1651
Figure 11-46. ADCCTL1 Register....1655
Figure 11-47. ADCCTL2 Register....1657
Figure 11-48. ADCBURSTCTL Register....1658
Figure 11-49. ADCINTFLG Register....1660
Figure 11-50. ADCINTFLGCLR Register....1662
Figure 11-51. ADCINTOVF Register....1663
Figure 11-52. ADCINTOVFCLR Register....1664
Figure 11-53. ADCINTSEL1N2 Register....1665
Figure 11-54. ADCINTSEL3N4 Register....1667
Figure 11-55. ADCSOCPRICTL Register....1669
Figure 11-56. ADCINTSOCSEL1 Register....1671
Figure 11-57. ADCINTSOCSEL2 Register....1673
Figure 11-58. ADCSOCFLG1 Register....1675
Figure 11-59. ADCSOCFRC1 Register....1679
Figure 11-60. ADCSOCOVF1 Register....1684
Figure 11-61. ADCSOCOVFCLR1 Register....1687
Figure 11-62. ADCSOC0CTL Register....1690
Figure 11-63. ADCSOC1CTL Register....1693
Figure 11-64. ADCSOC2CTL Register....1696
Figure 11-65. ADCSOC3CTL Register....1699
Figure 11-66. ADCSOC4CTL Register....1702
Figure 11-67. ADCSOC5CTL Register....1705
Figure 11-68. ADCSOC6CTL Register....1708
Figure 11-69. ADCSOC7CTL Register....1711
Figure 11-70. ADCSOC8CTL Register....1714
Figure 11-71. ADCSOC9CTL Register....1717
Figure 11-72. ADCSOC10CTL Register....1720
Figure 11-73. ADCSOC11CTL Register....1723
Figure 11-74. ADCSOC12CTL Register....1726
Figure 11-75. ADCSOC13CTL Register....1729
Figure 11-76. ADCSOC14CTL Register....1732
Figure 11-77. ADCSOC15CTL Register....1735
Figure 11-78. ADCEVTSTAT Register....1738
Figure 11-79. ADCEVTCLR Register....1741
Figure 11-80. ADCEVTSEL Register....1743
Figure 11-81. ADCEVTINTSEL Register....1745
Figure 11-82. ADCOSDETECT Register....1747
Figure 11-83. ADCCOUNTER Register....1748
Figure 11-84. ADCREV Register....1749
Figure 11-85. ADCOFFTRIM Register....1750
Figure 11-86. ADCPPB1CONFIG Register....1751
Figure 11-87. ADCPPB1STAMP Register....1753
Figure 11-88. ADCPPB1OFFCAL Register....1754
Figure 11-89. ADCPPB1OFFREF Register....1755
Figure 11-90. ADCPPB1TRIPHI Register....1756
Figure 11-91. ADCPPB1TRIPLO Register....1757
Figure 11-92. ADCPPB2CONFIG Register....1758
Figure 11-93. ADCPPB2STAMP Register....1760
Figure 11-94. ADCPPB2OFFCAL Register....1761
Figure 11-95. ADCPPB2OFFREF Register....1762
Figure 11-96. ADCPPB2TRIPHI Register....1763
Figure 11-97. ADCPPB2TRIPLO Register....1764
Figure 11-98. ADCPPB3CONFIG Register....1765
Figure 11-99. ADCPPB3STAMP Register....1767
Figure 11-100. ADCPPB3OFFCAL Register....1768
Figure 11-101. ADCPPB3OFFREF Register....1769



Copyright © 2024 Texas Instruments Incorporated

Figure 11-102. ADCPPB3TRIPHI Register....1770
Figure 11-103. ADCPPB3TRIPLO Register....1771
Figure 11-104. ADCPPB4CONFIG Register....1772
Figure 11-105. ADCPPB4STAMP Register....1774
Figure 11-106. ADCPPB4OFFCAL Register....1775
Figure 11-107. ADCPPB4OFFREF Register....1776
Figure 11-108. ADCPPB4TRIPHI Register....1777
Figure 11-109. ADCPPB4TRIPLO Register....1778
Figure 11-110. ADCINLTRIM1 Register....1779
Figure 11-111. ADCINLTRIM2 Register....1780
Figure 11-112. ADCINLTRIM3 Register....1781
Figure 11-113. ADCINLTRIM4 Register....1782
Figure 11-114. ADCINLTRIM5 Register....1783
Figure 11-115. ADCINLTRIM6 Register....1784
Figure 12-1. DAC Module Block Diagram....1791
Figure 12-2. DACREV Register....1795
Figure 12-3. DACCTL Register....1796
Figure 12-4. DACVALA Register....1797
Figure 12-5. DACVALS Register....1798
Figure 12-6. DACOUTEN Register....1799
Figure 12-7. DACLOCK Register....1800
Figure 12-8. DACTRIM Register....1801
Figure 13-1. CMPSS Module Block Diagram....1805
Figure 13-2. Comparator Block Diagram....1805
Figure 13-3. Reference DAC Block Diagram....1806
Figure 13-4. Ramp Generator Block Diagram....1808
Figure 13-5. Ramp Generator Behavior....1809
Figure 13-6. Digital Filter Behavior....1810
Figure 13-7. COMPCTL Register....1817
Figure 13-8. COMPHYSCTL Register....1819
Figure 13-9. COMPSTS Register....1820
Figure 13-10. COMPSTSCLR Register....1821
Figure 13-11. COMPDACCTL Register....1822
Figure 13-12. DACHVALS Register....1823
Figure 13-13. DACHVALA Register....1824
Figure 13-14. RAMPMAXREFA Register....1825
Figure 13-15. RAMPMAXREFS Register....1826
Figure 13-16. RAMPDECVALA Register....1827
Figure 13-17. RAMPDECVALS Register....1828
Figure 13-18. RAMPSTS Register....1829
Figure 13-19. DACLVALS Register....1830
Figure 13-20. DACLVALA Register....1831
Figure 13-21. RAMPDLYA Register....1832
Figure 13-22. RAMPDLYS Register....1833
Figure 13-23. CTRIPLFILCTL Register....1834
Figure 13-24. CTRIPLFILCLKCTL Register....1835
Figure 13-25. CTRIPHFILCTL Register....1836
Figure 13-26. CTRIPHFILCLKCTL Register....1837
Figure 13-27. COMPLOCK Register....1838
Figure 14-1. Sigma Delta Filter Module (SDFM) CPU Interface....1842
Figure 14-2. Sigma Delta Filter Module (SDFM) Block Diagram....1844
Figure 14-3. Block Diagram of One Filter Module....1845
Figure 14-4. Different Modulator Modes Supported....1848
Figure 14-5. Simplified Sinc Filter Architecture....1849
Figure 14-6. Z-Transform of Sinc Filter of Order N....1849
Figure 14-7. Frequency Response of Different Sinc Filters....1849
Figure 14-8. SDSYNC Event....1853
Figure 14-9. Comparator Unit Structure....1854
Figure 14-10. SDFM Error (SD_ERR) Interrupt Sources....1858
Figure 14-11. SDIFLG Register....1865
Figure 14-12. SDIFLGCLR Register....1867



Copyright © 2024 Texas Instruments Incorporated

Figure 14-13. SDCTL Register ... 1869
Figure 14-14. SDMFILEN Register ... 1870
Figure 14-15. SDCTLPARM1 Register ... 1871
Figure 14-16. SDDFPARM1 Register ... 1872
Figure 14-17. SDDPARM1 Register ... 1873
Figure 14-18. SDCMPH1 Register ... 1874
Figure 14-19. SDCMPL1 Register ... 1875
Figure 14-20. SDCPARM1 Register ... 1876
Figure 14-21. SDDATA1 Register ... 1877
Figure 14-22. SDCTLPARM2 Register ... 1878
Figure 14-23. SDDFPARM2 Register ... 1879
Figure 14-24. SDDPARM2 Register ... 1880
Figure 14-25. SDCMPH2 Register ... 1881
Figure 14-26. SDCMPL2 Register ... 1882
Figure 14-27. SDCPARM2 Register ... 1883
Figure 14-28. SDDATA2 Register ... 1884
Figure 14-29. SDCTLPARM3 Register ... 1885
Figure 14-30. SDDFPARM3 Register ... 1886
Figure 14-31. SDDPARM3 Register ... 1887
Figure 14-32. SDCMPH3 Register ... 1888
Figure 14-33. SDCMPL3 Register ... 1889
Figure 14-34. SDCPARM3 Register ... 1890
Figure 14-35. SDDATA3 Register ... 1891
Figure 14-36. SDCTLPARM4 Register ... 1892
Figure 14-37. SDDFPARM4 Register ... 1893
Figure 14-38. SDDPARM4 Register ... 1894
Figure 14-39. SDCMPH4 Register ... 1895
Figure 14-40. SDCMPL4 Register ... 1896
Figure 14-41. SDCPARM4 Register ... 1897
Figure 14-42. SDDATA4 Register ... 1898
Figure 15-1. Multiple ePWM Modules ... 1905
Figure 15-2. Submodules and Signal Connections for an ePWM Module ... 1906
Figure 15-3. ePWM Modules and Critical Internal Signal Interconnects ... 1908
Figure 15-4. Time-Base Submodule ... 1911
Figure 15-5. Time-Base Submodule Signals and Registers ... 1912
Figure 15-6. Time-Base Frequency and Period ... 1914
Figure 15-7. Time-Base Counter Synchronization Scheme ... 1916
Figure 15-8. Time-Base Up-Count Mode Waveforms ... 1918
Figure 15-9. Time-Base Down-Count Mode Waveforms ... 1919
Figure 15-10. Time-Base Up-Down-Count Waveforms, TBCTL[PHSDIR = 0] Count Down On Synchronization Event ... 1920
Figure 15-11. Time-Base Up-Down Count Waveforms, TBCTL[PHSDIR = 1] Count Up On Synchronization Event ... 1921
Figure 15-12. Global Load: Signals and Registers ... 1922
Figure 15-13. One-Shot Sync Mode ... 1923
Figure 15-14. Counter-Compare Submodule ... 1924
Figure 15-15. Detailed View of the Counter-Compare Submodule ... 1925
Figure 15-16. Counter-Compare Event Waveforms in Up-Count Mode ... 1928
Figure 15-17. Counter-Compare Events in Down-Count Mode ... 1928
Figure 15-18. Counter-Compare Events In Up-Down-Count Mode, TBCTL[PHSDIR = 0] Count Down On Synchronization Event ... 1929
Figure 15-19. Counter-Compare Events In Up-Down-Count Mode, TBCTL[PHSDIR = 1] Count Up On Synchronization Event ... 1929
Figure 15-20. Action-Qualifier Submodule ... 1930
Figure 15-21. Action-Qualifier Submodule Inputs and Outputs ... 1931
Figure 15-22. Possible Action-Qualifier Actions for EPWMxA and EPWMxB Outputs ... 1932
Figure 15-23. AQCTL[SHDWAQAMODE] ... 1935
Figure 15-24. AQCTL[SHDWAQBMODE] ... 1935
Figure 15-25. Up-Down Count Mode Symmetrical Waveform ... 1937
Figure 15-26. Up, Single Edge Asymmetric Waveform, with Independent Modulation on EPWMxA and EPWMxB—Active High ... 1938
Figure 15-27. Up, Single Edge Asymmetric Waveform with Independent Modulation on EPWMxA and EPWMxB—Active Low ... 1939



Copyright © 2024 Texas Instruments Incorporated

Figure 15-28. Up-Count, Pulse Placement Asymmetric Waveform With Independent Modulation on EPWMxA....1940
Figure 15-29. Up-Down Count, Dual-Edge Symmetric Waveform, with Independent Modulation on EPWMxA and EPWMxB — Active Low....1940
Figure 15-30. Up-Down Count, Dual-Edge Symmetric Waveform, with Independent Modulation on EPWMxA and EPWMxB — Complementary....1941
Figure 15-31. Up-Down Count, Dual-Edge Asymmetric Waveform, with Independent Modulation on EPWMxA—Active Low....1941
Figure 15-32. Up-Down Count, PWM Waveform Generation Utilizing T1 and T2 Events....1942
Figure 15-33. Dead_Band Submodule....1943
Figure 15-34. Configuration Options for the Dead-Band Submodule....1946
Figure 15-35. Dead-Band Waveforms for Typical Cases (0% < Duty < 100%)....1948
Figure 15-36. PWM Chopper Submodule....1950
Figure 15-37. PWM Chopper Submodule Operational Details....1951
Figure 15-38. Simple PWM Chopper Submodule Waveforms Showing Chopping Action Only....1951
Figure 15-39. PWM Chopper Submodule Waveforms Showing the First Pulse and Subsequent Sustaining Pulses....1952
Figure 15-40. PWM Chopper Submodule Waveforms Showing the Pulse Width (Duty Cycle) Control of Sustaining Pulses 1953
Figure 15-41. Trip-Zone Submodule....1954
Figure 15-42. Trip-Zone Submodule Mode Control Logic....1958
Figure 15-43. Trip-Zone Submodule Interrupt Logic....1959
Figure 15-44. Event-Trigger Submodule....1960
Figure 15-45. Event-Trigger Submodule Showing Event Inputs and Prescaled Outputs....1961
Figure 15-46. Event-Trigger Interrupt Generator....1963
Figure 15-47. Event-Trigger SOCA Pulse Generator....1964
Figure 15-48. Event-Trigger SOCB Pulse Generator....1964
Figure 15-49. Digital-Compare Submodule High-Level Block Diagram....1965
Figure 15-50. GPIO MUX-to-Trip Input Connectivity....1966
Figure 15-51. DCAEVT1 Event Triggering....1969
Figure 15-52. DCAEVT2 Event Triggering....1969
Figure 15-53. DCBEVT1 Event Triggering....1970
Figure 15-54. DCBEVT2 Event Triggering....1970
Figure 15-55. Event Filtering....1971
Figure 15-56. Blanking Window Timing Diagram....1972
Figure 15-57. Valley Switching....1974
Figure 15-58. ePWM X-BAR....1975
Figure 15-59. Simplified ePWM Module....1976
Figure 15-60. EPWM1 Configured as a Typical Master, EPWM2 Configured as a Slave....1977
Figure 15-61. Control of Four Buck Stages. Here $F_{PWM1} \neq F_{PWM2} \neq F_{PWM3} \neq F_{PWM4}$....1978
Figure 15-62. Buck Waveforms for Control of Four Buck Stages (Note: Only three bucks shown here)....1979
Figure 15-63. Control of Four Buck Stages. (Note: $F_{PWM2} = N \times F_{PWM1}$)....1980
Figure 15-64. Buck Waveforms for Control of Four Buck Stages (Note: $F_{PWM2} = F_{PWM1}$)....1981
Figure 15-65. Control of Two Half-H Bridge Stages ($F_{PWM2} = N \times F_{PWM1}$)....1982
Figure 15-66. Half-H Bridge Waveforms for Control of Two Half-H Bridge Stages (Note: Here $F_{PWM2} = F_{PWM1}$)....1983
Figure 15-67. Control of Dual 3-Phase Inverter Stages as Is Commonly Used in Motor Control....1984
Figure 15-68. 3-Phase Inverter Waveforms for Control of Dual 3-Phase Inverter Stages (Only One Inverter Shown)....1985
Figure 15-69. Configuring Two PWM Modules for Phase Control....1986
Figure 15-70. Timing Waveforms Associated with Phase Control Between Two Modules....1987
Figure 15-71. Control of 3-Phase Interleaved DC/DC Converter....1988
Figure 15-72. 3-Phase Interleaved DC/DC Converter Waveforms for Control of 3-Phase Interleaved DC/DC Converter....1989
Figure 15-73. Control of Full-H Bridge Stage ($F_{PWM2} = F_{PWM1}$)....1990
Figure 15-74. ZVS Full-H Bridge Waveforms....1991
Figure 15-75. Peak Current Mode Control of Buck Converter....1992
Figure 15-76. Peak Current Mode Control Waveforms for Control of Buck Converter....1992
Figure 15-77. Control of Two Resonant Converter Stages....1993
Figure 15-78. H-Bridge LLC Resonant Converter PWM Waveforms....1993
Figure 15-79. HRPWM Block Diagram....1994
Figure 15-80. Resolution Calculations for Conventionally Generated PWM....1995
Figure 15-81. Operating Logic Using MEP....1996
Figure 15-82. HRPWM Extension Registers and Memory Configuration....1997
Figure 15-83. HRPWM System Interface....1998
Figure 15-84. HRPWM and HRCAL Source Clock....1999
Figure 15-85. Required PWM Waveform for a Requested Duty = 40.5%....2002



Copyright © 2024 Texas Instruments Incorporated

Figure 15-86. Low % Duty Cycle Range Limitation Example (HRPCTL[HRPE] = 0) 2005
Figure 15-87. High % Duty Cycle Range Limitation Example (HRPCTL[HRPE] = 0) 2006
Figure 15-88. Up-Count Duty Cycle Range Limitation Example (HRPCTL[HRPE] = 1) 2006
Figure 15-89. Up-Down Count Duty Cycle Range Limitation Example (HRPCTL[HRPE] = 1) 2006
Figure 15-90. Simple Buck Controlled Converter Using a Single PWM 2013
Figure 15-91. PWM Waveform Generated for Simple Buck Controlled Converter 2013
Figure 15-92. Simple Reconstruction Filter for a PWM-based DAC 2015
Figure 15-93. PWM Waveform Generated for the PWM DAC Function 2015
Figure 15-94. TBCTL Register 2024
Figure 15-95. TBCTL2 Register 2026
Figure 15-96. TBCTR Register 2027
Figure 15-97. TBSTS Register 2028
Figure 15-98. CMPCTL Register 2029
Figure 15-99. CMPCTL2 Register 2031
Figure 15-100. DBCTL Register 2033
Figure 15-101. DBCTL2 Register 2036
Figure 15-102. AQCTL Register 2037
Figure 15-103. AQTSRCSEL Register 2039
Figure 15-104. PCCTL Register 2040
Figure 15-105. VCAPCTL Register 2042
Figure 15-106. VCNTCFG Register 2044
Figure 15-107. HRCNFG Register 2046
Figure 15-108. HRPWR Register 2048
Figure 15-109. HRMSTEP Register 2049
Figure 15-110. HRCNFG2 Register 2050
Figure 15-111. HRPCTL Register 2051
Figure 15-112. TRREM Register 2053
Figure 15-113. GLDCTL Register 2054
Figure 15-114. GLDCFG Register 2056
Figure 15-115. EPWMXLINK Register 2058
Figure 15-116. AQCTLA Register 2060
Figure 15-117. AQCTLA2 Register 2062
Figure 15-118. AQCTLB Register 2063
Figure 15-119. AQCTLB2 Register 2065
Figure 15-120. AQSFRC Register 2066
Figure 15-121. AQCSFRC Register 2067
Figure 15-122. DBREDHR Register 2068
Figure 15-123. DBRED Register 2069
Figure 15-124. DBFEDHR Register 2070
Figure 15-125. DBFED Register 2071
Figure 15-126. TBPHS Register 2072
Figure 15-127. TBPRDHR Register 2073
Figure 15-128. TBPRD Register 2074
Figure 15-129. CMPA Register 2075
Figure 15-130. CMPB Register 2076
Figure 15-131. CMPC Register 2077
Figure 15-132. CMPD Register 2078
Figure 15-133. GLDCTL2 Register 2079
Figure 15-134. SWVDELVAL Register 2080
Figure 15-135. TZSEL Register 2081
Figure 15-136. TZDCSEL Register 2083
Figure 15-137. TZCTL Register 2084
Figure 15-138. TZCTL2 Register 2086
Figure 15-139. TZCTLDCA Register 2088
Figure 15-140. TZCTLDCB Register 2090
Figure 15-141. TZEINT Register 2092
Figure 15-142. TZFLG Register 2093
Figure 15-143. TZCBCFLG Register 2095
Figure 15-144. TZOSTFLG Register 2097
Figure 15-145. TZCLR Register 2099
Figure 15-146. TZCBCCLR Register 2101



Copyright © 2024 Texas Instruments Incorporated

Figure 15-147. TZOSTCLR Register....2102
Figure 15-148. TZFRC Register....2103
Figure 15-149. ETSEL Register....2104
Figure 15-150. ETPS Register....2107
Figure 15-151. ETFLG Register....2110
Figure 15-152. ETCLR Register....2111
Figure 15-153. ETFRC Register....2112
Figure 15-154. ETINTPS Register....2113
Figure 15-155. ETSOCPS Register....2114
Figure 15-156. ETCNTINITCTL Register....2116
Figure 15-157. ETCNTINIT Register....2117
Figure 15-158. DCTRIPSEL Register....2118
Figure 15-159. DCACTL Register....2120
Figure 15-160. DCBCTL Register....2121
Figure 15-161. DCFCTL Register....2122
Figure 15-162. DCCAPCTL Register....2124
Figure 15-163. DCFOFFSET Register....2126
Figure 15-164. DCFOFFSETCNT Register....2127
Figure 15-165. DCFWINDOW Register....2128
Figure 15-166. DCFWINDOWCNT Register....2129
Figure 15-167. DCCAP Register....2130
Figure 15-168. DCAHTRIPSEL Register....2131
Figure 15-169. DCALTRIPSEL Register....2133
Figure 15-170. DCBHTRIPSEL Register....2135
Figure 15-171. DCBLTRIPSEL Register....2137
Figure 15-172. HWVDELVAL Register....2139
Figure 15-173. VCNTVAL Register....2140
Figure 16-1. Capture and APWM Modes of Operation....2156
Figure 16-2. Counter Compare and PRD Effects on the eCAP Output in APWM Mode....2157
Figure 16-3. eCAP Block Diagram....2158
Figure 16-4. Event Prescale Control....2159
Figure 16-5. Prescale Function Waveforms....2159
Figure 16-6. Details of the Continuous/One-shot Block....2160
Figure 16-7. Details of the Counter and Synchronization Block....2162
Figure 16-8. Time-Base Counter Synchronization Scheme....2163
Figure 16-9. Interrupts in eCAP Module....2164
Figure 16-10. PWM Waveform Details Of APWM Mode Operation....2165
Figure 16-11. Time-Base Frequency and Period Calculation....2166
Figure 16-12. Capture Sequence for Absolute Time-stamp and Rising-Edge Detect....2167
Figure 16-13. Capture Sequence for Absolute Time-stamp with Rising- and Falling-Edge Detect....2168
Figure 16-14. Capture Sequence for Delta Mode Time-stamp and Rising Edge Detect....2169
Figure 16-15. Capture Sequence for Delta Mode Time-stamp with Rising- and Falling-Edge Detect....2170
Figure 16-16. PWM Waveform Details of APWM Mode Operation....2171
Figure 16-17. TSCTR Register....2175
Figure 16-18. CTRPHS Register....2176
Figure 16-19. CAP1 Register....2177
Figure 16-20. CAP2 Register....2178
Figure 16-21. CAP3 Register....2179
Figure 16-22. CAP4 Register....2180
Figure 16-23. ECCTL1 Register....2181
Figure 16-24. ECCTL2 Register....2183
Figure 16-25. ECEINT Register....2185
Figure 16-26. ECFLG Register....2187
Figure 16-27. ECCLR Register....2188
Figure 16-28. ECFRC Register....2189
Figure 17-1. Optical Encoder Disk....2193
Figure 17-2. QEP Encoder Output Signal for Forward/Reverse Movement....2193
Figure 17-3. Index Pulse Example....2194
Figure 17-4. Functional Block Diagram of the eQEP Peripheral....2197
Figure 17-5. Functional Block Diagram of Decoder Unit....2199
Figure 17-6. Quadrature Decoder State Machine....2200



Copyright © 2024 Texas Instruments Incorporated

Figure 17-7. Quadrature-clock and Direction Decoding ..... 2201
Figure 17-8. Position Counter Reset by Index Pulse for 1000-Line Encoder (QPOSMAX = 3999 or 0xF9F) ..... 2203
Figure 17-9. Position Counter Underflow/Overflow (QPOSMAX = 4) ..... 2204
Figure 17-10. Software Index Marker for 1000-line Encoder (QEPCTL[IEL] = 1) ..... 2206
Figure 17-11. Strobe Event Latch (QEPCTL[SEL] = 1) ..... 2206
Figure 17-12. eQEP Position-compare Unit ..... 2208
Figure 17-13. eQEP Position-compare Event Generation Points ..... 2209
Figure 17-14. eQEP Position-compare Sync Output Pulse Stretcher ..... 2209
Figure 17-15. eQEP Edge Capture Unit ..... 2211
Figure 17-16. Unit Position Event for Low Speed Measurement (QCAPCTL[UPPS] = 0010) ..... 2212
Figure 17-17. eQEP Edge Capture Unit - Timing Details ..... 2212
Figure 17-18. eQEP Watchdog Timer ..... 2214
Figure 17-19. eQEP Unit Timer Base ..... 2214
Figure 17-20. eQEP Interrupt Generation ..... 2215
Figure 17-21. QPOSCNT Register ..... 2217
Figure 17-22. QPOSINIT Register ..... 2218
Figure 17-23. QPOSMAX Register ..... 2219
Figure 17-24. QPOSCMP Register ..... 2220
Figure 17-25. QPOSILAT Register ..... 2221
Figure 17-26. QPOSSLAT Register ..... 2222
Figure 17-27. QPOSLAT Register ..... 2223
Figure 17-28. QUTMR Register ..... 2224
Figure 17-29. QUPRD Register ..... 2225
Figure 17-30. QWDTMR Register ..... 2226
Figure 17-31. QWDPRD Register ..... 2227
Figure 17-32. QDECCTL Register ..... 2228
Figure 17-33. QEPCTL Register ..... 2230
Figure 17-34. QCAPCTL Register ..... 2232
Figure 17-35. QPOSCTL Register ..... 2233
Figure 17-36. QEINT Register ..... 2234
Figure 17-37. QFLG Register ..... 2236
Figure 17-38. QCLR Register ..... 2238
Figure 17-39. QFRC Register ..... 2240
Figure 17-40. QEPSTS Register ..... 2242
Figure 17-41. QCTMR Register ..... 2244
Figure 17-42. QCPRD Register ..... 2245
Figure 17-43. QCTMRLAT Register ..... 2246
Figure 17-44. QCPRDLAT Register ..... 2247
Figure 18-1. SPI CPU Interface ..... 2252
Figure 18-2. SPI Interrupt Flags and Enable Logic Generation ..... 2255
Figure 18-3. SPI DMA Trigger Diagram ..... 2256
Figure 18-4. SPI Master/Slave Connection ..... 2257
Figure 18-5. SPI Module Master Configuration ..... 2259
Figure 18-6. SPI Module Slave Configuration ..... 2260
Figure 18-7. SPICLK Signal Options ..... 2263
Figure 18-8. SPI: SPICLK-LSPCLK Characteristic when (BRR + 1) is Odd, BRR > 3, and CLKPOLARITY = 1 ..... 2264
Figure 18-9. SPI 3-wire Master Mode ..... 2266
Figure 18-10. SPI 3-wire Slave Mode ..... 2267
Figure 18-11. Five Bits per Character ..... 2270
Figure 18-12. SPI Digital Audio Receiver Configuration Using Two SPIs ..... 2273
Figure 18-13. Standard Right-Justified Digital Audio Data Format ..... 2273
Figure 18-14. SPICCR Register ..... 2279
Figure 18-15. SPICTL Register ..... 2281
Figure 18-16. SPISTS Register ..... 2283
Figure 18-17. SPIBRR Register ..... 2285
Figure 18-18. SPIRXEMU Register ..... 2286
Figure 18-19. SPIRXBUF Register ..... 2287
Figure 18-20. SPITXBUF Register ..... 2288
Figure 18-21. SPIDAT Register ..... 2289
Figure 18-22. SPIFFTX Register ..... 2290
Figure 18-23. SPIFFRX Register ..... 2292



Copyright © 2024 Texas Instruments Incorporated

Figure 18-24. SPIFFCT Register....2294
Figure 18-25. SPIPRI Register....2295
Figure 19-1. SCI CPU Interface....2299
Figure 19-2. Serial Communications Interface (SCI) Module Block Diagram....2301
Figure 19-3. Typical SCI Data Frame Formats....2303
Figure 19-4. Idle-Line Multiprocessor Communication Format....2305
Figure 19-5. Double-Buffered WUT and TXSHF....2306
Figure 19-6. Address-Bit Multiprocessor Communication Format....2307
Figure 19-7. SCI Asynchronous Communications Format....2308
Figure 19-8. SCI RX Signals in Communication Modes....2309
Figure 19-9. SCI TX Signals in Communications Mode....2310
Figure 19-10. SCI FIFO Interrupt Flags and Enable Logic....2314
Figure 19-11. SCICCR Register....2318
Figure 19-12. SCICTL1 Register....2320
Figure 19-13. SCIHBAUD Register....2322
Figure 19-14. SCILBAUD Register....2323
Figure 19-15. SCICTL2 Register....2324
Figure 19-16. SCIRXST Register....2326
Figure 19-17. SCIRXEMU Register....2329
Figure 19-18. SCIRXBUF Register....2330
Figure 19-19. SCITXBUF Register....2332
Figure 19-20. SCIFFTX Register....2333
Figure 19-21. SCIFFRX Register....2335
Figure 19-22. SCIFFCT Register....2337
Figure 19-23. SCIPRI Register....2338
Figure 20-1. Multiple I2C Modules Connected....2342
Figure 20-2. I2C Module Conceptual Block Diagram....2345
Figure 20-3. Clocking Diagram for the I2C Module....2345
Figure 20-4. Roles of the Clock Divide-Down Values (ICCL and ICCH)....2346
Figure 20-5. Bit Transfer on the I2C bus....2347
Figure 20-6. I2C Slave TX / RX Flowchart....2349
Figure 20-7. I2C Master TX / RX Flowchart....2350
Figure 20-8. I2C Module START and STOP Conditions....2351
Figure 20-9. I2C Module Data Transfer (7-Bit Addressing with 8-bit Data Configuration Shown)....2352
Figure 20-10. I2C Module 7-Bit Addressing Format (FDF = 0, XA = 0 in I2CMDR)....2353
Figure 20-11. I2C Module 10-Bit Addressing Format (FDF = 0, XA = 1 in I2CMDR)....2353
Figure 20-12. I2C Module Free Data Format (FDF = 1 in I2CMDR)....2354
Figure 20-13. Repeated START Condition (in This Case, 7-Bit Addressing Format)....2354
Figure 20-14. Synchronization of Two I2C Clock Generators During Arbitration....2355
Figure 20-15. Arbitration Procedure Between Two Master-Transmitters....2356
Figure 20-16. Pin Diagram Showing the Effects of the Digital Loopback Mode (DLB) Bit....2357
Figure 20-17. Enable Paths of the I2C Interrupt Requests....2360
Figure 20-18. Backwards Compatibility Mode and Forward Compatibility Bit, Slave Transmitter....2361
Figure 20-19. I2C FIFO Interrupt....2362
Figure 20-20. I2COAR Register....2367
Figure 20-21. I2CIER Register....2368
Figure 20-22. I2CSTR Register....2369
Figure 20-23. I2CCLKL Register....2373
Figure 20-24. I2CCLKH Register....2374
Figure 20-25. I2CCNT Register....2375
Figure 20-26. I2CDRR Register....2376
Figure 20-27. I2CSAR Register....2377
Figure 20-28. I2CDXR Register....2378
Figure 20-29. I2CMDR Register....2379
Figure 20-30. I2CISRC Register....2383
Figure 20-31. I2CEMDR Register....2384
Figure 20-32. I2CPSC Register....2385
Figure 20-33. I2CFFTX Register....2386
Figure 20-34. I2CFFRX Register....2388
Figure 21-1. Conceptual Block Diagram of the McBSP....2394
Figure 21-2. McBSP Data Transfer Paths....2395



Copyright © 2024 Texas Instruments Incorporated

Figure 21-3. Companding Processes ..... 2396
Figure 21-4. µ-Law Transmit Data Companding Format ..... 2396
Figure 21-5. A-Law Transmit Data Companding Format ..... 2396
Figure 21-6. Two Methods by Which the McBSP Can Compand Internal Data ..... 2397
Figure 21-7. Example - Clock Signal Control of Bit Transfer Timing ..... 2397
Figure 21-8. McBSP Operating at Maximum Packet Frequency ..... 2399
Figure 21-9. Single-Phase Frame for a McBSP Data Transfer ..... 2400
Figure 21-10. Dual-Phase Frame for a McBSP Data Transfer ..... 2400
Figure 21-11. Implementing the AC97 Standard With a Dual-Phase Frame ..... 2401
Figure 21-12. Timing of an AC97-Standard Data Transfer Near Frame Synchronization ..... 2401
Figure 21-13. McBSP Reception Physical Data Path ..... 2402
Figure 21-14. McBSP Reception Signal Activity ..... 2402
Figure 21-15. McBSP Transmission Physical Data Path ..... 2403
Figure 21-16. McBSP Transmission Signal Activity ..... 2403
Figure 21-17. Conceptual Block Diagram of the Sample Rate Generator ..... 2405
Figure 21-18. Possible Inputs to the Sample Rate Generator and the Polarity Bits ..... 2407
Figure 21-19. CLKG Synchronization and FSG Generation When GSYNC = 1 and CLKGDV = 1 ..... 2409
Figure 21-20. CLKG Synchronization and FSG Generation When GSYNC = 1 and CLKGDV = 3 ..... 2410
Figure 21-21. Overrun in the McBSP Receiver ..... 2412
Figure 21-22. Overrun Prevented in the McBSP Receiver ..... 2413
Figure 21-23. Possible Responses to Receive Frame-Synchronization Pulses ..... 2413
Figure 21-24. An Unexpected Frame-Synchronization Pulse During a McBSP Reception ..... 2414
Figure 21-25. Proper Positioning of Frame-Synchronization Pulses ..... 2415
Figure 21-26. Data in the McBSP Transmitter Overwritten and Thus Not Transmitted ..... 2415
Figure 21-27. Underflow During McBSP Transmission ..... 2416
Figure 21-28. Underflow Prevented in the McBSP Transmitter ..... 2417
Figure 21-29. Possible Responses to Transmit Frame-Synchronization Pulses ..... 2417
Figure 21-30. An Unexpected Frame-Synchronization Pulse During a McBSP Transmission ..... 2418
Figure 21-31. Proper Positioning of Frame-Synchronization Pulses ..... 2419
Figure 21-32. Alternating Between the Channels of Partition A and the Channels of Partition B ..... 2422
Figure 21-33. Reassigning Channel Blocks Throughout a McBSP Data Transfer ..... 2422
Figure 21-34. McBSP Data Transfer in the 8-Partition Mode ..... 2423
Figure 21-35. Activity on McBSP Pins for the Possible Values of XMCM ..... 2427
Figure 21-36. Typical SPI Interface ..... 2428
Figure 21-37. SPI Transfer with CLKSTP = 10b (No Clock Delay), CLKXP = 0, and CLKRP = 0 ..... 2430
Figure 21-38. SPI Transfer with CLKSTP = 11b (Clock Delay), CLKXP = 0, CLKRP = 1 ..... 2430
Figure 21-39. SPI Transfer with CLKSTP = 10b (No Clock Delay), CLKXP = 1, and CLKRP = 0 ..... 2431
Figure 21-40. SPI Transfer with CLKSTP = 11b (Clock Delay), CLKXP = 1, CLKRP = 1 ..... 2431
Figure 21-41. SPI Interface with McBSP Used as Master ..... 2433
Figure 21-42. SPI Interface With McBSP Used as Slave ..... 2434
Figure 21-43. Unexpected Frame-Synchronization Pulse With (R/X)FIG = 0 ..... 2442
Figure 21-44. Unexpected Frame-Synchronization Pulse With (R/X)FIG = 1 ..... 2442
Figure 21-45. Companding Processes for Reception and for Transmission ..... 2443
Figure 21-46. Range of Programmable Data Delay ..... 2444
Figure 21-47. 2-Bit Data Delay Used to Skip a Framing Bit ..... 2445
Figure 21-48. Data Clocked Externally Using a Rising Edge and Sampled by the McBSP Receiver on a Falling Edge ..... 2450
Figure 21-49. Frame of Period 16 CLKG Periods and Active Width of 2 CLKG Periods ..... 2451
Figure 21-50. Data Clocked Externally Using a Rising Edge and Sampled by the McBSP Receiver on a Falling Edge ..... 2454
Figure 21-51. Unexpected Frame-Synchronization Pulse With (R/X) FIG = 0 ..... 2465
Figure 21-52. Unexpected Frame-Synchronization Pulse With (R/X) FIG = 1 ..... 2465
Figure 21-53. Companding Processes for Reception and for Transmission ..... 2466
Figure 21-54. µ-Law Transmit Data Companding Format ..... 2467
Figure 21-55. A-Law Transmit Data Companding Format ..... 2467
Figure 21-56. Range of Programmable Data Delay ..... 2468
Figure 21-57. 2-Bit Data Delay Used to Skip a Framing Bit ..... 2468
Figure 21-58. Data Clocked Externally Using a Rising Edge and Sampled by the McBSP Receiver on a Falling Edge ..... 2472
Figure 21-59. Frame of Period 16 CLKG Periods and Active Width of 2 CLKG Periods ..... 2472
Figure 21-60. Data Clocked Externally Using a Rising Edge and Sampled by the McBSP Receiver on a Falling Edge ..... 2474
Figure 21-61. Four 8-Bit Data Words Transferred To/From the McBSP ..... 2478
Figure 21-62. One 32-Bit Data Word Transferred To/From the McBSP ..... 2479
Figure 21-63. 8-Bit Data Words Transferred at Maximum Packet Frequency ..... 2480



Copyright © 2024 Texas Instruments Incorporated

Figure 21-64. Configuring the Data Stream of Figure 21-63 as a Continuous 32-Bit Word....2480
Figure 21-65. Receive Interrupt Generation....2481
Figure 21-66. Transmit Interrupt Generation....2482
Figure 21-67. DRR2 Register....2488
Figure 21-68. DRR1 Register....2489
Figure 21-69. DXR2 Register....2490
Figure 21-70. DXR1 Register....2491
Figure 21-71. SPCR2 Register....2492
Figure 21-72. SPCR1 Register....2495
Figure 21-73. RCR2 Register....2498
Figure 21-74. RCR1 Register....2500
Figure 21-75. XCR2 Register....2501
Figure 21-76. XCR1 Register....2503
Figure 21-77. SRGR2 Register....2504
Figure 21-78. SRGR1 Register....2506
Figure 21-79. MCR2 Register....2507
Figure 21-80. MCR1 Register....2509
Figure 21-81. RCERA Register....2511
Figure 21-82. RCERB Register....2512
Figure 21-83. XCERA Register....2513
Figure 21-84. XCERB Register....2514
Figure 21-85. PCR Register....2515
Figure 21-86. RCERC Register....2518
Figure 21-87. RCERD Register....2519
Figure 21-88. XCERC Register....2520
Figure 21-89. XCERD Register....2521
Figure 21-90. RCERE Register....2522
Figure 21-91. RCERF Register....2523
Figure 21-92. XCERE Register....2524
Figure 21-93. XCERF Register....2525
Figure 21-94. RCERG Register....2526
Figure 21-95. RCERH Register....2527
Figure 21-96. XCERG Register....2528
Figure 21-97. XCERH Register....2529
Figure 21-98. MFFINT Register....2530
Figure 22-1. CAN Block Diagram....2536
Figure 22-2. Accessing Message Objects Through IFx Registers....2537
Figure 22-3. CAN_MUX....2542
Figure 22-4. CAN Core in Silent Mode....2543
Figure 22-5. CAN Core in Loopback Mode....2544
Figure 22-6. CAN Core in External Loopback Mode....2545
Figure 22-7. CAN Core in Loopback Combined with Silent Mode....2546
Figure 22-8. CAN Interrupt Topology 1....2548
Figure 22-9. CAN Interrupt Topology 2....2548
Figure 22-10. Initialization of a Transmit Object....2551
Figure 22-11. Initialization of a Single Receive Object for Data Frames....2551
Figure 22-12. Initialization of a Single Receive Object for Remote Frames....2552
Figure 22-13. CPU Handling of a FIFO Buffer (Interrupt Driven)....2557
Figure 22-14. Bit Timing....2558
Figure 22-15. Propagation Time Segment....2559
Figure 22-16. Synchronization on Late and Early Edges....2561
Figure 22-17. Filtering of Short Dominant Spikes....2562
Figure 22-18. Structure of the CAN Core's CAN Protocol Controller....2564
Figure 22-19. Data Transfer Between IF1 / IF2 Registers and Message RAM....2568
Figure 22-20. Structure of a Message Object....2569
Figure 22-21. Message RAM Representation in Debug Mode....2573
Figure 22-22. CAN_CTL Register....2577
Figure 22-23. CAN_ES Register....2580
Figure 22-24. CAN_ERRC Register....2582
Figure 22-25. CAN_BTR Register....2583
Figure 22-26. CAN_INT Register....2585



Copyright © 2024 Texas Instruments Incorporated

Figure 22-27. CAN_TEST Register ..... 2586
Figure 22-28. CAN_PERR Register ..... 2588
Figure 22-29. CAN_RAM_INIT Register ..... 2589
Figure 22-30. CAN_GLB_INT_EN Register ..... 2590
Figure 22-31. CAN_GLB_INT_FLG Register ..... 2591
Figure 22-32. CAN_GLB_INT_CLR Register ..... 2592
Figure 22-33. CAN_ABOTR Register ..... 2593
Figure 22-34. CAN_TXRQ_X Register ..... 2594
Figure 22-35. CAN_TXRQ_21 Register ..... 2595
Figure 22-36. CAN_NDAT_X Register ..... 2596
Figure 22-37. CAN_NDAT_21 Register ..... 2597
Figure 22-38. CAN_IPEN_X Register ..... 2598
Figure 22-39. CAN_IPEN_21 Register ..... 2599
Figure 22-40. CAN_MVAL_X Register ..... 2600
Figure 22-41. CAN_MVAL_21 Register ..... 2601
Figure 22-42. CAN_IP_MUX21 Register ..... 2602
Figure 22-43. CAN_IF1CMD Register ..... 2603
Figure 22-44. CAN_IF1MSK Register ..... 2606
Figure 22-45. CAN_IF1ARB Register ..... 2607
Figure 22-46. CAN_IF1MCTL Register ..... 2609
Figure 22-47. CAN_IF1DATA Register ..... 2611
Figure 22-48. CAN_IF1DATB Register ..... 2612
Figure 22-49. CAN_IF2CMD Register ..... 2613
Figure 22-50. CAN_IF2MSK Register ..... 2616
Figure 22-51. CAN_IF2ARB Register ..... 2617
Figure 22-52. CAN_IF2MCTL Register ..... 2619
Figure 22-53. CAN_IF2DATA Register ..... 2621
Figure 22-54. CAN_IF2DATB Register ..... 2622
Figure 22-55. CAN_IF3OBS Register ..... 2623
Figure 22-56. CAN_IF3MSK Register ..... 2625
Figure 22-57. CAN_IF3ARB Register ..... 2626
Figure 22-58. CAN_IF3MCTL Register ..... 2627
Figure 22-59. CAN_IF3DATA Register ..... 2629
Figure 22-60. CAN_IF3DATB Register ..... 2630
Figure 22-61. CAN_IF3UPD Register ..... 2631
Figure 23-1. USB Block Diagram ..... 2637
Figure 23-2. USB Scheme ..... 2638
Figure 23-3. USB Device Functional Address Register (USBFADDR) ..... 2660
Figure 23-4. USB Power Management Register (USBPOWER) in Host Mode ..... 2661
Figure 23-5. USB Power Management Register (USBPOWER) in Device Mode ..... 2662
Figure 23-6. USB Transmit Interrupt Status Register (USBTXIS) ..... 2663
Figure 23-7. USB Transmit Interrupt Status Register (USBRXIS) ..... 2665
Figure 23-8. USB Transmit Interrupt Status Enable Register (USBTXIE) ..... 2667
Figure 23-9. USB Transmit Interrupt Status Enable Register (USBRXIE) ..... 2669
Figure 23-10. USB General Interrupt Status Register (USBIS) in Host Mode ..... 2671
Figure 23-11. USB General Interrupt Status Register (USBIS) in Device Mode ..... 2672
Figure 23-12. USB Interrupt Enable Register (USBIE) in Host Mode ..... 2673
Figure 23-13. USB Interrupt Enable Register (USBIE) in Device Mode ..... 2674
Figure 23-14. USB Frame Value Register (USBFRAME) ..... 2675
Figure 23-15. USB Endpoint Index Register (USBEPIDX) ..... 2675
Figure 23-16. USB Test Mode Register (USBTEST) in Host Mode ..... 2676
Figure 23-17. USB Test Mode Register (USBTEST) in Device Mode ..... 2677
Figure 23-18. USB FIFO Endpoint *n* Register (USBFIFO[*n*]) ..... 2678
Figure 23-19. USB Device Control Register (USBDEVCTL) ..... 2679
Figure 23-20. USB Transmit Dynamic FIFO Sizing Register (USBTXFIFOSZ) ..... 2681
Figure 23-21. USB Receive Dynamic FIFO Sizing Register (USBRXFIFOSZ) ..... 2682
Figure 23-22. USB Transmit FIFO Start Address Register (USBTXFIFOADDR]) ..... 2683
Figure 23-23. USB Receive FIFO Start Address Register (USBRXFIFOADDR) ..... 2684
Figure 23-24. USB Connect Timing Register (USBCONTIM) ..... 2685
Figure 23-25. USB Full-Speed Last Transaction to End of Frame Timing Register (USBFSEOF) ..... 2686
Figure 23-26. USB Low-Speed Last Transaction to End of Frame Timing Register (USBLSEOF) ..... 2686



Copyright © 2024 Texas Instruments Incorporated

Figure 23-27. USB Transmit Functional Address Endpoint *n* Registers (USBTXFUNCADDR[*n*]) 2687
Figure 23-28. USB Transmit Hub Address Endpoint *n* Registers (USBTXHUBADDR[*n*]) 2688
Figure 23-29. USB Transmit Hub Port Endpoint *n* Registers (USBTXHUBPORT[*n*]) 2689
Figure 23-30. USB Receive Functional Address Endpoint *n* Registers (USBFIFO[*n*]) 2690
Figure 23-31. USB Receive Hub Address Endpoint *n* Registers (USBRXHUBADDR[*n*]) 2691
Figure 23-32. USB Receive Hub Port Endpoint *n* Register (USBRXHUBPORT[*n*]) 2692
Figure 23-33. USB Control and Status Endpoint 0 Low Register (USBCSRL0) in Host Mode 2693
Figure 23-34. USB Control and Status Endpoint 0 Low Register (USBCSRL0) in Device Mode 2694
Figure 23-35. USB Control and Status Endpoint 0 High Register (USBCSRH0) in Host Mode 2696
Figure 23-36. USB Control and Status Endpoint 0 High Register (USBCSRH0) in Device Mode 2697
Figure 23-37. USB Receive Byte Count Endpoint 0 Register (USBCOUNT0) 2698
Figure 23-38. USB Type Endpoint 0 Register (USBTYPE0) 2698
Figure 23-39. USB NAK Limit Register (USBNAKLMT) 2699
Figure 23-40. USB Maximum Transmit Data Endpoint *n* Registers (USBTXMAXP[*n*]) 2700
Figure 23-41. USB Transmit Control and Status Endpoint n Low Register (USBTXCSRL[*n*]) in Host Mode 2701
Figure 23-42. USB Transmit Control and Status Endpoint n Low Register (USBTXCSRL[*n*]) in Device Mode 2702
Figure 23-43. USB Transmit Control and Status Endpoint n High Register (USBTXCSRH[*n*]) in Host Mode 2704
Figure 23-44. USB Transmit Control and Status Endpoint n High Register (USBTXCSRH[*n*]) in Device Mode 2705
Figure 23-45. USB Maximum Receive Data Endpoint *n* Registers (USBRXMAXP[*n*]) 2706
Figure 23-46. USB Receive Control and Status Endpoint n Low Register (USBRXCSRL[*n*]) in Host Mode 2707
Figure 23-47. USB Receive Control and Status Endpoint n Low Register (USBRXCSRL[*n*]) in Device Mode 2708
Figure 23-48. USB Receive Control and Status Endpoint n High Register (USBRXCSRH[*n*]) in Host Mode 2710
Figure 23-49. USB Receive Control and Status Endpoint n High Register (USBRXCSRH[*n*]) in Device Mode 2711
Figure 23-50. USB Receive Byte Count Endpoint n Register (USBRXCOUNT[*n*]) 2712
Figure 23-51. USB Host Transmit Configure Type Endpoint n Registers (USBTXTYPE[*n*]) 2713
Figure 23-52. USB Host Transmit Interval Endpoint *n* Registers (USBTXINTERVAL[*n*]) 2714
Figure 23-53. USB Host Configure Receive Type Endpoint n Register (USBRXTYPE[*n*]) 2715
Figure 23-54. USB Host Receive Polling Interval Endpoint *n* Registers (USBRXINTERVAL[*n*]) 2716
Figure 23-55. USB Request Packet Count in Block Transfer Endpoint *n* Registers (USBRQPKTCOUNT[*n*]) 2717
Figure 23-56. USB Receive Double Packet Buffer Disable Register (USBRXDPKTBUFDIS) 2718
Figure 23-57. USB Transmit Double Packet Buffer Disable Register (USBTXDPKTBUFDIS) 2719
Figure 23-58. USB External Power Control Register (USBEPC) 2720
Figure 23-59. USB External Power Control Raw Interrupt Status Register (USBEPCRIS) 2722
Figure 23-60. USB External Power Control Interrupt Mask Register (USBEPCIM) 2723
Figure 23-61. USB External Power Control Interrupt Status and Clear Register (USBEPCISC) 2724
Figure 23-62. USB Device RESUME Raw Interrupt Status Register (USBDRRIS) 2725
Figure 23-63. USB Device RESUME Raw Interrupt Status Register (USBDRRIS) 2726
Figure 23-64. USB Device RESUME Interrupt Status and Clear Register (USBDRISC) 2727
Figure 23-65. USB General-Purpose Control and Status Register (USBGPCS) 2728
Figure 23-66. USB DMA Select Register (USBDMASEL) 2729
Figure 24-1. uPP Integration 2749
Figure 24-2. Functional Block Diagram 2750
Figure 24-3. RX in SDR or DDR (Non-Demux) Mode 2751
Figure 24-4. RX in DDR (Demux) Mode 2751
Figure 24-5. TX in SDR (Non-Interleave) or DDR (Non-Demux) Mode 2751
Figure 24-6. TX in SDR (Interleave) or DDR (Demux) Mode 2751
Figure 24-7. IO Output Clock Generation for TX Mode 2752
Figure 24-8. IO Input Clock for RX Mode 2752
Figure 24-9. Structure of DMA Window and Lines in Memory 2754
Figure 24-10. uPP Receive in SDR Mode 2756
Figure 24-11. uPP Transmit in SDR Mode 2756
Figure 24-12. uPP Transmit in SDR Mode – Interleaving 2757
Figure 24-13. uPP Receive DDR Case 2757
Figure 24-14. uPP Transmit DDR Case 2757
Figure 24-15. uPP Tx Data Pattern in Non-Interleaved Mode 2758
Figure 24-16. uPP Rx Data Pattern in Non-Interleaved Mode 2758
Figure 24-17. PID Register 2766
Figure 24-18. PERCTL Register 2767
Figure 24-19. CHCTL Register 2769
Figure 24-20. IFCFG Register 2770
Figure 24-21. IFIVAL Register 2772



Copyright © 2024 Texas Instruments Incorporated

Figure 24-22. THCFG Register....2773
Figure 24-23. RAWINTST Register....2775
Figure 24-24. ENINTST Register....2777
Figure 24-25. INTENSET Register....2779
Figure 24-26. INTENCLR Register....2781
Figure 24-27. CHIDESC0 Register....2783
Figure 24-28. CHIDESC1 Register....2784
Figure 24-29. CHIDESC2 Register....2785
Figure 24-30. CHIST0 Register....2786
Figure 24-31. CHIST1 Register....2787
Figure 24-32. CHIST2 Register....2788
Figure 24-33. CHQDESC0 Register....2789
Figure 24-34. CHQDESC1 Register....2790
Figure 24-35. CHQDESC2 Register....2791
Figure 24-36. CHQST0 Register....2792
Figure 24-37. CHQST1 Register....2793
Figure 24-38. CHQST2 Register....2794
Figure 24-39. GINTEN Register....2795
Figure 24-40. GINTFLG Register....2796
Figure 24-41. GINTCLR Register....2797
Figure 24-42. DLYCTL Register....2798
Figure 25-1. EMIF Module Overview....2802
Figure 25-2. EMIF Functional Block Diagram....2805
Figure 25-3. Timing Waveform of SDRAM PRE Command....2809
Figure 25-4. EMIF to 2M × 16 × 4 Bank SDRAM Interface....2810
Figure 25-5. EMIF to 512K × 16 × 2 Bank SDRAM Interface....2810
Figure 25-6. Timing Waveform for Basic SDRAM Read Operation....2818
Figure 25-7. Timing Waveform for Basic SDRAM Write Operation....2819
Figure 25-8. EMIF Asynchronous Interface....2821
Figure 25-9. EMIF to 8-bit/16-bit Memory Interface....2822
Figure 25-10. Common Asynchronous Interface....2822
Figure 25-11. Timing Waveform of an Asynchronous Read Cycle in Normal Mode....2826
Figure 25-12. Timing Waveform of an Asynchronous Write Cycle in Normal Mode....2828
Figure 25-13. Timing Waveform of an Asynchronous Read Cycle in Select Strobe Mode....2830
Figure 25-14. Timing Waveform of an Asynchronous Write Cycle in Select Strobe Mode....2832
Figure 25-15. Example Configuration Interface....2837
Figure 25-16. SDRAM Timing Register (SDRAM_TR)....2838
Figure 25-17. SDRAM Self Refresh Exit Timing Register (SDR_EXT_TMNG)....2839
Figure 25-18. SDRAM Refresh Control Register (SDRAM_RCR)....2839
Figure 25-19. SDRAM Configuration Register (SDRAM_CR)....2840
Figure 25-20. LH28F800BJE-PTTL90 to EMIF Read Timing Waveforms....2841
Figure 25-21. LH28F800BJE-PTTL90 to EMIF Write Timing Waveforms....2842
Figure 25-22. Asynchronous *m* Configuration Register (*m* = 1, 2) (ASYNC_CS*n*_CR(*n* = 2, 3))....2844
Figure 25-23. RCSR Register....2846
Figure 25-24. ASYNC_WCCR Register....2847
Figure 25-25. SDRAM_CR Register....2848
Figure 25-26. SDRAM_RCR Register....2850
Figure 25-27. ASYNC_CS2_CR Register....2851
Figure 25-28. ASYNC_CS3_CR Register....2853
Figure 25-29. ASYNC_CS4_CR Register....2855
Figure 25-30. SDRAM_TR Register....2857
Figure 25-31. TOTAL_SDRAM_AR Register....2858
Figure 25-32. TOTAL_SDRAM_ACTR Register....2859
Figure 25-33. SDR_EXT_TMNG Register....2860
Figure 25-34. INT_RAW Register....2861
Figure 25-35. INT_MSK Register....2862
Figure 25-36. INT_MSK_SET Register....2863
Figure 25-37. INT_MSK_CLR Register....2864
Figure 25-38. EMIF1LOCK Register....2866
Figure 25-39. EMIF1COMMIT Register....2867
Figure 25-40. EMIF1MSEL Register....2868



Copyright © 2024 Texas Instruments Incorporated

Figure 25-41. EMIF1ACCPROT0 Register....2869
Figure 25-42. EMIF2LOCK Register....2871
Figure 25-43. EMIF2COMMIT Register....2872
Figure 25-44. EMIF2ACCPROT0 Register....2873
Figure 26-1. Block Diagram of the CLB Subsystem in the Device....2877
Figure 26-2. Block Diagram of a CLB Tile and CPU Interface....2877
Figure 26-3. CLB Clocking....2878
Figure 26-4. GPIO to CLB Tile Connections....2879
Figure 26-5. CLB Input Mux and Filter....2880
Figure 26-6. CLB Input Synchronization Example....2880
Figure 26-7. CLB Outputs....2885
Figure 26-8. CLB Output Signal Multiplexer....2886
Figure 26-9. CLB Tile Submodules....2887
Figure 26-10. Counter Block....2890
Figure 26-11. FSM Block....2893
Figure 26-12. FSM LUT Block....2894
Figure 26-13. LUT4 Block....2895
Figure 26-14. Output LUT Block....2895
Figure 26-15. High Level Controller Block....2896
Figure 26-16. CLB_COUNT_RESET Register....2908
Figure 26-17. CLB_COUNT_MODE_1 Register....2909
Figure 26-18. CLB_COUNT_MODE_0 Register....2910
Figure 26-19. CLB_COUNT_EVENT Register....2911
Figure 26-20. CLB_FSM_EXTRA_IN0 Register....2912
Figure 26-21. CLB_FSM_EXTERNAL_IN0 Register....2913
Figure 26-22. CLB_FSM_EXTERNAL_IN1 Register....2914
Figure 26-23. CLB_FSM_EXTRA_IN1 Register....2915
Figure 26-24. CLB_LUT4_IN0 Register....2916
Figure 26-25. CLB_LUT4_IN1 Register....2917
Figure 26-26. CLB_LUT4_IN2 Register....2918
Figure 26-27. CLB_LUT4_IN3 Register....2919
Figure 26-28. CLB_FSM_LUT_FN1_0 Register....2920
Figure 26-29. CLB_FSM_LUT_FN2 Register....2921
Figure 26-30. CLB_LUT4_FN1_0 Register....2922
Figure 26-31. CLB_LUT4_FN2 Register....2923
Figure 26-32. CLB_FSM_NEXT_STATE_0 Register....2924
Figure 26-33. CLB_FSM_NEXT_STATE_1 Register....2925
Figure 26-34. CLB_FSM_NEXT_STATE_2 Register....2926
Figure 26-35. CLB_MISC_CONTROL Register....2927
Figure 26-36. CLB_OUTPUT_LUT_0 Register....2929
Figure 26-37. CLB_OUTPUT_LUT_1 Register....2930
Figure 26-38. CLB_OUTPUT_LUT_2 Register....2931
Figure 26-39. CLB_OUTPUT_LUT_3 Register....2932
Figure 26-40. CLB_OUTPUT_LUT_4 Register....2933
Figure 26-41. CLB_OUTPUT_LUT_5 Register....2934
Figure 26-42. CLB_OUTPUT_LUT_6 Register....2935
Figure 26-43. CLB_OUTPUT_LUT_7 Register....2936
Figure 26-44. CLB_HLC_EVENT_SEL Register....2937
Figure 26-45. CLB_LOAD_EN Register....2940
Figure 26-46. CLB_LOAD_ADDR Register....2941
Figure 26-47. CLB_LOAD_DATA Register....2942
Figure 26-48. CLB_INPUT_FILTER Register....2943
Figure 26-49. CLB_IN_MUX_SEL_0 Register....2945
Figure 26-50. CLB_LCL_MUX_SEL_1 Register....2947
Figure 26-51. CLB_LCL_MUX_SEL_2 Register....2948
Figure 26-52. CLB_BUF_PTR Register....2949
Figure 26-53. CLB_GP_REG Register....2950
Figure 26-54. CLB_OUT_EN Register....2951
Figure 26-55. CLB_GLBL_MUX_SEL_1 Register....2952
Figure 26-56. CLB_GLBL_MUX_SEL_2 Register....2953
Figure 26-57. CLB_INTR_TAG_REG Register....2954



Copyright © 2024 Texas Instruments Incorporated

Figure 26-58. CLB_LOCK Register ..... 2955
Figure 26-59. CLB_DBG_R0 Register ..... 2956
Figure 26-60. CLB_DBG_R1 Register ..... 2957
Figure 26-61. CLB_DBG_R2 Register ..... 2958
Figure 26-62. CLB_DBG_R3 Register ..... 2959
Figure 26-63. CLB_DBG_C0 Register ..... 2960
Figure 26-64. CLB_DBG_C1 Register ..... 2961
Figure 26-65. CLB_DBG_C2 Register ..... 2962
Figure 26-66. CLB_DBG_OUT Register ..... 2963
Figure 26-67. CLB_PUSH_y Register ..... 2966
Figure 26-68. CLB_PULL_y Register ..... 2967

## List of Tables

Table 1-1. C2000Ware Root Directories ..... 76
Table 2-1. TMU Supported Instructions ..... 80
Table 2-2. Viterbi Decode Performance ..... 81
Table 2-3. Complex Math Performance ..... 81
Table 3-1. Reset Signals ..... 84
Table 3-2. PIE Channel Mapping ..... 92
Table 3-3. CPU Interrupt Vectors ..... 94
Table 3-4. PIE Interrupt Vectors ..... 95
Table 3-5. Access to EALLOW-Protected Registers ..... 101
Table 3-6. Clock Connections Sorted by Clock Domain ..... 110
Table 3-7. Clock Connections Sorted by Module Name ..... 111
Table 3-8. Clock Source (OSCCLK) Failure Detection ..... 114
Table 3-9. Example Watchdog Key Sequences ..... 119
Table 3-10. LPM Entry and Exit Criteria ..... 123
Table 3-11. Local Shared RAM ..... 126
Table 3-12. Global Shared RAM ..... 126
Table 3-13. Error Handling in Different Scenarios ..... 131
Table 3-14. Mapping of ECC Bits in Read Data from ECC/Parity Address Map ..... 132
Table 3-15. Mapping of Parity Bits in Read Data from ECC/Parity Address Map ..... 132
Table 3-16. CLA Access Filter ..... 147
Table 3-17. RAM Status ..... 148
Table 3-18. Security Levels ..... 148
Table 3-19. System Control Registers Impacted ..... 160
Table 3-20. System Control Base Address Table ..... 166
Table 3-21. CPUTIMER_REGS Registers ..... 167
Table 3-22. CPUTIMER_REGS Access Type Codes ..... 167
Table 3-23. TIM Register Field Descriptions ..... 168
Table 3-24. PRD Register Field Descriptions ..... 169
Table 3-25. TCR Register Field Descriptions ..... 170
Table 3-26. TPR Register Field Descriptions ..... 172
Table 3-27. TPRH Register Field Descriptions ..... 173
Table 3-28. PIE_CTRL_REGS Registers ..... 174
Table 3-29. PIE_CTRL_REGS Access Type Codes ..... 174
Table 3-30. PIECTRL Register Field Descriptions ..... 176
Table 3-31. PIEACK Register Field Descriptions ..... 177
Table 3-32. PIEIER1 Register Field Descriptions ..... 178
Table 3-33. PIEIFR1 Register Field Descriptions ..... 180
Table 3-34. PIEIER2 Register Field Descriptions ..... 182
Table 3-35. PIEIFR2 Register Field Descriptions ..... 184
Table 3-36. PIEIER3 Register Field Descriptions ..... 186
Table 3-37. PIEIFR3 Register Field Descriptions ..... 188
Table 3-38. PIEIER4 Register Field Descriptions ..... 190
Table 3-39. PIEIFR4 Register Field Descriptions ..... 192
Table 3-40. PIEIER5 Register Field Descriptions ..... 194
Table 3-41. PIEIFR5 Register Field Descriptions ..... 196
Table 3-42. PIEIER6 Register Field Descriptions ..... 198
Table 3-43. PIEIFR6 Register Field Descriptions ..... 200



Copyright © 2024 Texas Instruments Incorporated

Table 3-44. PIEIER7 Register Field Descriptions....202
Table 3-45. PIEIFR7 Register Field Descriptions....204
Table 3-46. PIEIER8 Register Field Descriptions....206
Table 3-47. PIEIFR8 Register Field Descriptions....208
Table 3-48. PIEIER9 Register Field Descriptions....210
Table 3-49. PIEIFR9 Register Field Descriptions....212
Table 3-50. PIEIER10 Register Field Descriptions....214
Table 3-51. PIEIFR10 Register Field Descriptions....216
Table 3-52. PIEIER11 Register Field Descriptions....218
Table 3-53. PIEIFR11 Register Field Descriptions....220
Table 3-54. PIEIER12 Register Field Descriptions....222
Table 3-55. PIEIFR12 Register Field Descriptions....224
Table 3-56. WD_REGS Registers....226
Table 3-57. WD_REGS Access Type Codes....226
Table 3-58. SCSR Register Field Descriptions....227
Table 3-59. WDCNTR Register Field Descriptions....228
Table 3-60. WDKEY Register Field Descriptions....229
Table 3-61. WDCR Register Field Descriptions....230
Table 3-62. WDWCR Register Field Descriptions....231
Table 3-63. NMI_INTRUPT_REGS Registers....232
Table 3-64. NMI_INTRUPT_REGS Access Type Codes....232
Table 3-65. NMICFG Register Field Descriptions....233
Table 3-66. NMIFLG Register Field Descriptions....234
Table 3-67. NMIFLGCLR Register Field Descriptions....236
Table 3-68. NMIFLGFRC Register Field Descriptions....239
Table 3-69. NMIWDCNT Register Field Descriptions....241
Table 3-70. NMIWDPRD Register Field Descriptions....242
Table 3-71. NMISHDFLG Register Field Descriptions....243
Table 3-72. XINT_REGS Registers....245
Table 3-73. XINT_REGS Access Type Codes....245
Table 3-74. XINT1CR Register Field Descriptions....246
Table 3-75. XINT2CR Register Field Descriptions....247
Table 3-76. XINT3CR Register Field Descriptions....248
Table 3-77. XINT4CR Register Field Descriptions....249
Table 3-78. XINT5CR Register Field Descriptions....250
Table 3-79. XINT1CTR Register Field Descriptions....251
Table 3-80. XINT2CTR Register Field Descriptions....252
Table 3-81. XINT3CTR Register Field Descriptions....253
Table 3-82. SYNC_SOC_REGS Registers....254
Table 3-83. SYNC_SOC_REGS Access Type Codes....254
Table 3-84. SYNCSELECT Register Field Descriptions....255
Table 3-85. ADCSOCOUTSELECT Register Field Descriptions....257
Table 3-86. SYNCSOCLOCK Register Field Descriptions....260
Table 3-87. DMA_CLA_SRC_SEL_REGS Registers....261
Table 3-88. DMA_CLA_SRC_SEL_REGS Access Type Codes....261
Table 3-89. CLA1TASKSRCSELLOCK Register Field Descriptions....262
Table 3-90. DMACHSRCSELLOCK Register Field Descriptions....263
Table 3-91. CLA1TASKSRCSEL1 Register Field Descriptions....264
Table 3-92. CLA1TASKSRCSEL2 Register Field Descriptions....265
Table 3-93. DMACHSRCSEL1 Register Field Descriptions....266
Table 3-94. DMACHSRCSEL2 Register Field Descriptions....267
Table 3-95. FLASH_PUMP_SEMAPHORE_REGS Registers....268
Table 3-96. FLASH_PUMP_SEMAPHORE_REGS Access Type Codes....268
Table 3-97. PUMPREQUEST Register Field Descriptions....269
Table 3-98. DEV_CFG_REGS Registers....270
Table 3-99. DEV_CFG_REGS Access Type Codes....271
Table 3-100. DEVCFGLOCK1 Register Field Descriptions....272
Table 3-101. PARTIDL Register Field Descriptions....274
Table 3-102. PARTIDH Register Field Descriptions....276
Table 3-103. REVID Register Field Descriptions....277
Table 3-104. DC0 Register Field Descriptions....278



Copyright © 2024 Texas Instruments Incorporated

Table 3-105. DC1 Register Field Descriptions....279
Table 3-106. DC2 Register Field Descriptions....280
Table 3-107. DC3 Register Field Descriptions....281
Table 3-108. DC4 Register Field Descriptions....283
Table 3-109. DC5 Register Field Descriptions....284
Table 3-110. DC6 Register Field Descriptions....285
Table 3-111. DC7 Register Field Descriptions....286
Table 3-112. DC8 Register Field Descriptions....287
Table 3-113. DC9 Register Field Descriptions....288
Table 3-114. DC10 Register Field Descriptions....289
Table 3-115. DC11 Register Field Descriptions....290
Table 3-116. DC12 Register Field Descriptions....291
Table 3-117. DC13 Register Field Descriptions....292
Table 3-118. DC14 Register Field Descriptions....293
Table 3-119. DC15 Register Field Descriptions....294
Table 3-120. DC17 Register Field Descriptions....296
Table 3-121. DC18 Register Field Descriptions....297
Table 3-122. DC19 Register Field Descriptions....298
Table 3-123. DC20 Register Field Descriptions....299
Table 3-124. PERCNF1 Register Field Descriptions....301
Table 3-125. FUSEERR Register Field Descriptions....302
Table 3-126. SOFTPRES0 Register Field Descriptions....303
Table 3-127. SOFTPRES1 Register Field Descriptions....304
Table 3-128. SOFTPRES2 Register Field Descriptions....305
Table 3-129. SOFTPRES3 Register Field Descriptions....307
Table 3-130. SOFTPRES4 Register Field Descriptions....308
Table 3-131. SOFTPRES6 Register Field Descriptions....309
Table 3-132. SOFTPRES7 Register Field Descriptions....310
Table 3-133. SOFTPRES8 Register Field Descriptions....311
Table 3-134. SOFTPRES9 Register Field Descriptions....312
Table 3-135. SOFTPRES11 Register Field Descriptions....313
Table 3-136. SOFTPRES13 Register Field Descriptions....314
Table 3-137. SOFTPRES14 Register Field Descriptions....315
Table 3-138. SOFTPRES16 Register Field Descriptions....316
Table 3-139. CPUSEL0 Register Field Descriptions....317
Table 3-140. CPUSEL1 Register Field Descriptions....319
Table 3-141. CPUSEL2 Register Field Descriptions....320
Table 3-142. CPUSEL3 Register Field Descriptions....321
Table 3-143. CPUSEL4 Register Field Descriptions....322
Table 3-144. CPUSEL5 Register Field Descriptions....323
Table 3-145. CPUSEL6 Register Field Descriptions....324
Table 3-146. CPUSEL7 Register Field Descriptions....325
Table 3-147. CPUSEL8 Register Field Descriptions....326
Table 3-148. CPUSEL9 Register Field Descriptions....327
Table 3-149. CPUSEL11 Register Field Descriptions....328
Table 3-150. CPUSEL12 Register Field Descriptions....330
Table 3-151. CPUSEL14 Register Field Descriptions....332
Table 3-152. CPU2RESCTL Register Field Descriptions....333
Table 3-153. RSTSTAT Register Field Descriptions....334
Table 3-154. LPMSTAT Register Field Descriptions....335
Table 3-155. SYSDBGCTL Register Field Descriptions....336
Table 3-156. CLK_CFG_REGS Registers....337
Table 3-157. CLK_CFG_REGS Access Type Codes....337
Table 3-158. CLKSEM Register Field Descriptions....339
Table 3-159. CLKCFGLOCK1 Register Field Descriptions....340
Table 3-160. CLKSRCCTL1 Register Field Descriptions....342
Table 3-161. CLKSRCCTL2 Register Field Descriptions....344
Table 3-162. CLKSRCCTL3 Register Field Descriptions....346
Table 3-163. SYSPLLCTL1 Register Field Descriptions....347
Table 3-164. SYSPLLMULT Register Field Descriptions....348
Table 3-165. SYSPLLSTS Register Field Descriptions....349



Copyright © 2024 Texas Instruments Incorporated

Table 3-166. AUXPLLCTL1 Register Field Descriptions..... 350
Table 3-167. AUXPLLMULT Register Field Descriptions..... 351
Table 3-168. AUXPLLSTS Register Field Descriptions..... 352
Table 3-169. SYSCLKDIVSEL Register Field Descriptions..... 353
Table 3-170. AUXCLKDIVSEL Register Field Descriptions..... 354
Table 3-171. PERCLKDIVSEL Register Field Descriptions..... 355
Table 3-172. XCLKOUTDIVSEL Register Field Descriptions..... 356
Table 3-173. LOSPCP Register Field Descriptions..... 357
Table 3-174. MCDCR Register Field Descriptions..... 358
Table 3-175. X1CNT Register Field Descriptions..... 359
Table 3-176. CPU_SYS_REGS Registers..... 360
Table 3-177. CPU_SYS_REGS Access Type Codes..... 360
Table 3-178. CPUSYSLOCK1 Register Field Descriptions..... 362
Table 3-179. HIBBOOTMODE Register Field Descriptions..... 365
Table 3-180. IORESTOREADDR Register Field Descriptions..... 366
Table 3-181. PIEVERRADDR Register Field Descriptions..... 367
Table 3-182. PCLKCR0 Register Field Descriptions..... 368
Table 3-183. PCLKCR1 Register Field Descriptions..... 370
Table 3-184. PCLKCR2 Register Field Descriptions..... 371
Table 3-185. PCLKCR3 Register Field Descriptions..... 373
Table 3-186. PCLKCR4 Register Field Descriptions..... 374
Table 3-187. PCLKCR6 Register Field Descriptions..... 375
Table 3-188. PCLKCR7 Register Field Descriptions..... 376
Table 3-189. PCLKCR8 Register Field Descriptions..... 377
Table 3-190. PCLKCR9 Register Field Descriptions..... 378
Table 3-191. PCLKCR10 Register Field Descriptions..... 379
Table 3-192. PCLKCR11 Register Field Descriptions..... 380
Table 3-193. PCLKCR12 Register Field Descriptions..... 381
Table 3-194. PCLKCR13 Register Field Descriptions..... 382
Table 3-195. PCLKCR14 Register Field Descriptions..... 383
Table 3-196. PCLKCR16 Register Field Descriptions..... 385
Table 3-197. SECMSEL Register Field Descriptions..... 386
Table 3-198. LPMCR Register Field Descriptions..... 387
Table 3-199. GPIOLPMSEL0 Register Field Descriptions..... 389
Table 3-200. GPIOLPMSEL1 Register Field Descriptions..... 392
Table 3-201. TMR2CLKCTL Register Field Descriptions..... 395
Table 3-202. RESC Register Field Descriptions..... 397
Table 3-203. ROM_PREFETCH_REGS Registers..... 399
Table 3-204. ROM_PREFETCH_REGS Access Type Codes..... 399
Table 3-205. ROMPREFETCH Register Field Descriptions..... 400
Table 3-206. DCSM_Z1_REGS Registers..... 401
Table 3-207. DCSM_Z1_REGS Access Type Codes..... 401
Table 3-208. Z1_LINKPOINTER Register Field Descriptions..... 402
Table 3-209. Z1_OTPSECLOCK Register Field Descriptions..... 403
Table 3-210. Z1_BOOTCTRL Register Field Descriptions..... 404
Table 3-211. Z1_LINKPOINTERERR Register Field Descriptions..... 405
Table 3-212. Z1_CSMKEY0 Register Field Descriptions..... 406
Table 3-213. Z1_CSMKEY1 Register Field Descriptions..... 407
Table 3-214. Z1_CSMKEY2 Register Field Descriptions..... 408
Table 3-215. Z1_CSMKEY3 Register Field Descriptions..... 409
Table 3-216. Z1_CR Register Field Descriptions..... 410
Table 3-217. Z1_GRABSECTR Register Field Descriptions..... 411
Table 3-218. Z1_GRABRAMR Register Field Descriptions..... 414
Table 3-219. Z1_EXEONLYSECTR Register Field Descriptions..... 416
Table 3-220. Z1_EXEONLYRAMR Register Field Descriptions..... 419
Table 3-221. DCSM_Z2_REGS Registers..... 421
Table 3-222. DCSM_Z2_REGS Access Type Codes..... 421
Table 3-223. Z2_LINKPOINTER Register Field Descriptions..... 422
Table 3-224. Z2_OTPSECLOCK Register Field Descriptions..... 423
Table 3-225. Z2_BOOTCTRL Register Field Descriptions..... 424
Table 3-226. Z2_LINKPOINTERERR Register Field Descriptions..... 425



Copyright © 2024 Texas Instruments Incorporated

Table 3-227. Z2_CSMKEY0 Register Field Descriptions....426
Table 3-228. Z2_CSMKEY1 Register Field Descriptions....427
Table 3-229. Z2_CSMKEY2 Register Field Descriptions....428
Table 3-230. Z2_CSMKEY3 Register Field Descriptions....429
Table 3-231. Z2_CR Register Field Descriptions....430
Table 3-232. Z2_GRABSECTR Register Field Descriptions....431
Table 3-233. Z2_GRABRAMR Register Field Descriptions....434
Table 3-234. Z2_EXEONLYSECTR Register Field Descriptions....436
Table 3-235. Z2_EXEONLYRAMR Register Field Descriptions....439
Table 3-236. DCSM_COMMON_REGS Registers....441
Table 3-237. DCSM_COMMON_REGS Access Type Codes....441
Table 3-238. FLSEM Register Field Descriptions....442
Table 3-239. SECTSTAT Register Field Descriptions....443
Table 3-240. RAMSTAT Register Field Descriptions....446
Table 3-241. MEM_CFG_REGS Registers....448
Table 3-242. MEM_CFG_REGS Access Type Codes....448
Table 3-243. DxLOCK Register Field Descriptions....450
Table 3-244. DxCOMMIT Register Field Descriptions....451
Table 3-245. DxACCPROT0 Register Field Descriptions....452
Table 3-246. DxTEST Register Field Descriptions....453
Table 3-247. DxINIT Register Field Descriptions....454
Table 3-248. DxINITDONE Register Field Descriptions....455
Table 3-249. LSxLOCK Register Field Descriptions....456
Table 3-250. LSxCOMMIT Register Field Descriptions....458
Table 3-251. LSxMSEL Register Field Descriptions....460
Table 3-252. LSxCLAPGM Register Field Descriptions....462
Table 3-253. LSxACCPROT0 Register Field Descriptions....463
Table 3-254. LSxACCPROT1 Register Field Descriptions....465
Table 3-255. LSxTEST Register Field Descriptions....466
Table 3-256. LSxINIT Register Field Descriptions....468
Table 3-257. LSxINITDONE Register Field Descriptions....469
Table 3-258. GSxLOCK Register Field Descriptions....470
Table 3-259. GSxCOMMIT Register Field Descriptions....472
Table 3-260. GSxMSEL Register Field Descriptions....475
Table 3-261. GSxACCPROT0 Register Field Descriptions....477
Table 3-262. GSxACCPROT1 Register Field Descriptions....479
Table 3-263. GSxACCPROT2 Register Field Descriptions....481
Table 3-264. GSxACCPROT3 Register Field Descriptions....483
Table 3-265. GSxTEST Register Field Descriptions....485
Table 3-266. GSxINIT Register Field Descriptions....488
Table 3-267. GSxINITDONE Register Field Descriptions....490
Table 3-268. MSGxTEST Register Field Descriptions....492
Table 3-269. MSGxINIT Register Field Descriptions....493
Table 3-270. MSGxINITDONE Register Field Descriptions....494
Table 3-271. ACCESS_PROTECTION_REGS Registers....495
Table 3-272. ACCESS_PROTECTION_REGS Access Type Codes....495
Table 3-273. NMAVFLG Register Field Descriptions....497
Table 3-274. NMAVSET Register Field Descriptions....499
Table 3-275. NMAVCLR Register Field Descriptions....501
Table 3-276. NMAVINTEN Register Field Descriptions....503
Table 3-277. NMCPURDAVADDR Register Field Descriptions....504
Table 3-278. NMCPUWRAVADDR Register Field Descriptions....505
Table 3-279. NMCPUFAVADDR Register Field Descriptions....506
Table 3-280. NMDMAWRAVADDR Register Field Descriptions....507
Table 3-281. NMCLA1RDAVADDR Register Field Descriptions....508
Table 3-282. NMCLA1WRAVADDR Register Field Descriptions....509
Table 3-283. NMCLA1FAVADDR Register Field Descriptions....510
Table 3-284. MAVFLG Register Field Descriptions....511
Table 3-285. MAVSET Register Field Descriptions....512
Table 3-286. MAVCLR Register Field Descriptions....513
Table 3-287. MAVINTEN Register Field Descriptions....514



Copyright © 2024 Texas Instruments Incorporated

Table 3-288. MCPUFAVADDR Register Field Descriptions....515
Table 3-289. MCPUWRAVADDR Register Field Descriptions....516
Table 3-290. MDMAWRAVADDR Register Field Descriptions....517
Table 3-291. MEMORY_ERROR_REGS Registers....518
Table 3-292. MEMORY_ERROR_REGS Access Type Codes....518
Table 3-293. UCERRFLG Register Field Descriptions....519
Table 3-294. UCERRSET Register Field Descriptions....520
Table 3-295. UCERRCLR Register Field Descriptions....521
Table 3-296. UCCPUREADDR Register Field Descriptions....522
Table 3-297. UCDMAREADDR Register Field Descriptions....523
Table 3-298. UCCLA1READDR Register Field Descriptions....524
Table 3-299. CERRFLG Register Field Descriptions....525
Table 3-300. CERRSET Register Field Descriptions....526
Table 3-301. CERRCLR Register Field Descriptions....527
Table 3-302. CCPUREADDR Register Field Descriptions....528
Table 3-303. CERRCNT Register Field Descriptions....529
Table 3-304. CERRTHRES Register Field Descriptions....530
Table 3-305. CEINTFLG Register Field Descriptions....531
Table 3-306. CEINTCLR Register Field Descriptions....532
Table 3-307. CEINTSET Register Field Descriptions....533
Table 3-308. CEINTEN Register Field Descriptions....534
Table 3-309. ROM_WAIT_STATE_REGS Registers....535
Table 3-310. ROM_WAIT_STATE_REGS Access Type Codes....535
Table 3-311. ROMWAITSTATE Register Field Descriptions....536
Table 3-312. FLASH_CTRL_REGS Registers....537
Table 3-313. FLASH_CTRL_REGS Access Type Codes....537
Table 3-314. FRDCNTL Register Field Descriptions....538
Table 3-315. FBAC Register Field Descriptions....539
Table 3-316. FBFALLBACK Register Field Descriptions....540
Table 3-317. FBPRDY Register Field Descriptions....541
Table 3-318. FPAC1 Register Field Descriptions....542
Table 3-319. FMSTAT Register Field Descriptions....543
Table 3-320. FRD_INTF_CTRL Register Field Descriptions....545
Table 3-321. FLASH_ECC_REGS Registers....546
Table 3-322. FLASH_ECC_REGS Access Type Codes....546
Table 3-323. ECC_ENABLE Register Field Descriptions....548
Table 3-324. SINGLE_ERR_ADDR_LOW Register Field Descriptions....549
Table 3-325. SINGLE_ERR_ADDR_HIGH Register Field Descriptions....550
Table 3-326. UNC_ERR_ADDR_LOW Register Field Descriptions....551
Table 3-327. UNC_ERR_ADDR_HIGH Register Field Descriptions....552
Table 3-328. ERR_STATUS Register Field Descriptions....553
Table 3-329. ERR_POS Register Field Descriptions....555
Table 3-330. ERR_STATUS_CLR Register Field Descriptions....556
Table 3-331. ERR_CNT Register Field Descriptions....557
Table 3-332. ERR_THRESHOLD Register Field Descriptions....558
Table 3-333. ERR_INTFLG Register Field Descriptions....559
Table 3-334. ERR_INTCLR Register Field Descriptions....560
Table 3-335. FDATAH_TEST Register Field Descriptions....561
Table 3-336. FDATAL_TEST Register Field Descriptions....562
Table 3-337. FADDR_TEST Register Field Descriptions....563
Table 3-338. FECC_TEST Register Field Descriptions....564
Table 3-339. FECC_CTRL Register Field Descriptions....565
Table 3-340. FOUTH_TEST Register Field Descriptions....566
Table 3-341. FOUTL_TEST Register Field Descriptions....567
Table 3-342. FECC_STATUS Register Field Descriptions....568
Table 3-343. CPU_ID_REGS Registers....569
Table 3-344. CPU_ID_REGS Access Type Codes....569
Table 3-345. CPUID Register Field Descriptions....570
Table 3-346. UID_REGS Registers....571
Table 3-347. UID_REGS Access Type Codes....571
Table 3-348. UID_PSRAND0 Register Field Descriptions....572



Copyright © 2024 Texas Instruments Incorporated

Table 3-349. UID_PSRAND1 Register Field Descriptions....573
Table 3-350. UID_PSRAND2 Register Field Descriptions....574
Table 3-351. UID_PSRAND3 Register Field Descriptions....575
Table 3-352. UID_PSRAND4 Register Field Descriptions....576
Table 3-353. UID_PSRAND5 Register Field Descriptions....577
Table 3-354. UID_UNIQUE Register Field Descriptions....578
Table 3-355. UID_CHECKSUM Register Field Descriptions....579
Table 3-356. DCSM_Z1_OTP Registers....580
Table 3-357. DCSM_Z1_OTP Access Type Codes....580
Table 3-358. Z1OTP_LINKPOINTER1 Register Field Descriptions....581
Table 3-359. Z1OTP_LINKPOINTER2 Register Field Descriptions....582
Table 3-360. Z1OTP_LINKPOINTER3 Register Field Descriptions....583
Table 3-361. Z1OTP_PSWDLOCK Register Field Descriptions....584
Table 3-362. Z1OTP_CRCLOCK Register Field Descriptions....585
Table 3-363. Z1OTP_JTAGLOCK Register Field Descriptions....586
Table 3-364. Z1OTP_BOOTCTRL Register Field Descriptions....587
Table 3-365. DCSM_Z2_OTP Registers....588
Table 3-366. DCSM_Z2_OTP Access Type Codes....588
Table 3-367. Z2OTP_LINKPOINTER1 Register Field Descriptions....589
Table 3-368. Z2OTP_LINKPOINTER2 Register Field Descriptions....590
Table 3-369. Z2OTP_LINKPOINTER3 Register Field Descriptions....591
Table 3-370. Z2OTP_PSWDLOCK Register Field Descriptions....592
Table 3-371. Z2OTP_CRCLOCK Register Field Descriptions....593
Table 3-372. Z2OTP_JTAGLOCK Register Field Descriptions....594
Table 3-373. Z2OTP_BOOTCTRL Register Field Descriptions....595
Table 3-374. CPUTIMER Registers to Driverlib Functions....596
Table 3-375. ASYSCTL Registers to Driverlib Functions....596
Table 3-376. PIE Registers to Driverlib Functions....597
Table 3-377. SYSCTL Registers to Driverlib Functions....598
Table 3-378. NMI Registers to Driverlib Functions....605
Table 3-379. XINT Registers to Driverlib Functions....605
Table 3-380. DCSM Registers to Driverlib Functions....606
Table 3-381. MEMCFG Registers to Driverlib Functions....608
Table 3-382. FLASH Registers to Driverlib Functions....611
Table 4-1. ROM Memory....615
Table 4-2. Boot ROM Registers....615
Table 4-3. Boot ROM Sequence....615
Table 4-4. Device Default Boot Modes for CPU1....616
Table 4-5. All Available Boot Modes....616
Table 4-6. BOOTCTRL Register Bit Fields for CPU1....617
Table 4-7. BOOTCTRL Register Bit Fields for CPU2....617
Table 4-8. Get Mode Decoding on CPU1....619
Table 4-9. Get Mode Decoding on CPU2....619
Table 4-10. Emulation Boot Options....620
Table 4-11. Boot ROM Reset Causes and Actions....627
Table 4-12. Boot ROM Exceptions and Actions....628
Table 4-13. Entry Point Addresses for CPU1 and CPU2....629
Table 4-14. Wait Point Addresses for CPU1....629
Table 4-15. Wait Point Addresses for CPU2....629
Table 4-16. CPU1 Boot ROM Memory Map....630
Table 4-17. CPU2 Boot ROM Memory Map....630
Table 4-18. CLA Data ROM Memory Map....630
Table 4-19. Reserved RAM and Flash Memory-Map for CPU1....631
Table 4-20. Reserved RAM and Flash Memory-Map for CPU2....631
Table 4-21. CLA Data ROM Tables....632
Table 4-22. SPI 8-Bit Data Stream....636
Table 4-23. I2C 8-Bit Data Stream....640
Table 4-24. Parallel GPIO Boot 8-Bit Data Stream....641
Table 4-25. Bit-Rate Value for Internal Oscillators....645
Table 4-26. CAN 8-Bit Data Stream....646
Table 4-27. USB 8-Bit Data Stream....648



Copyright © 2024 Texas Instruments Incorporated

Table 4-28. LSB/MSB Loading Sequence in 8-Bit Data Stream....649
Table 4-29. SCI Boot Options....650
Table 4-30. CAN Boot Options....650
Table 4-31. I2C Boot Options....650
Table 4-32. USB Boot Options....651
Table 4-33. RAM Boot Options....651
Table 4-34. Flash Boot Options....651
Table 4-35. Wait Boot Options....651
Table 4-36. SPI Boot Options....651
Table 4-37. Parallel Boot Options....651
Table 4-38. Safe Copy Code Function....652
Table 4-39. Safe CRC Calculation Function....652
Table 4-40. C2TOC1IPC Commands Table....653
Table 4-41. C1TOC2IPC Commands Table....655
Table 4-42. CPU2 Error Command Values....658
Table 4-43. Boot Clock Sources....658
Table 4-44. Clock State After Boot ROM....658
Table 4-45. ROM Wait States....659
Table 4-46. CPU1 Boot Status Address....659
Table 4-47. CPU1 Boot Status Bit Fields....659
Table 4-48. CPU1 Boot Mode Status Address....660
Table 4-49. CPU1 Boot Mode Status Values....660
Table 4-50. CPU2 Boot Status Address....661
Table 4-51. CPU2 Boot Status Bit Fields....661
Table 4-52. CPU1 IPC NAK Status Bit Fields....661
Table 4-53. CPU2 IPC NAK Status Bit Fields....662
Table 4-54. Boot ROM Version Information for CPU1 and CPU2....662
Table 5-1. DMA Trigger Source Options....669
Table 5-2. BURSTSIZE versus DATASIZE Behavior....673
Table 5-3. DMA Base Address Table....682
Table 5-4. DMA_REGS Registers....683
Table 5-5. DMA_REGS Access Type Codes....683
Table 5-6. DMACTRL Register Field Descriptions....684
Table 5-7. DEBUGCTRL Register Field Descriptions....685
Table 5-8. PRIORITYCTRL1 Register Field Descriptions....686
Table 5-9. PRIORITYSTAT Register Field Descriptions....687
Table 5-10. DMA_CH_REGS Registers....688
Table 5-11. DMA_CH_REGS Access Type Codes....688
Table 5-12. MODE Register Field Descriptions....689
Table 5-13. CONTROL Register Field Descriptions....691
Table 5-14. BURST_SIZE Register Field Descriptions....693
Table 5-15. BURST_COUNT Register Field Descriptions....694
Table 5-16. SRC_BURST_STEP Register Field Descriptions....695
Table 5-17. DST_BURST_STEP Register Field Descriptions....696
Table 5-18. TRANSFER_SIZE Register Field Descriptions....697
Table 5-19. TRANSFER_COUNT Register Field Descriptions....698
Table 5-20. SRC_TRANSFER_STEP Register Field Descriptions....699
Table 5-21. DST_TRANSFER_STEP Register Field Descriptions....700
Table 5-22. SRC_WRAP_SIZE Register Field Descriptions....701
Table 5-23. SRC_WRAP_COUNT Register Field Descriptions....702
Table 5-24. SRC_WRAP_STEP Register Field Descriptions....703
Table 5-25. DST_WRAP_SIZE Register Field Descriptions....704
Table 5-26. DST_WRAP_COUNT Register Field Descriptions....705
Table 5-27. DST_WRAP_STEP Register Field Descriptions....706
Table 5-28. SRC_BEG_ADDR_SHADOW Register Field Descriptions....707
Table 5-29. SRC_ADDR_SHADOW Register Field Descriptions....708
Table 5-30. SRC_BEG_ADDR_ACTIVE Register Field Descriptions....709
Table 5-31. SRC_ADDR_ACTIVE Register Field Descriptions....710
Table 5-32. DST_BEG_ADDR_SHADOW Register Field Descriptions....711
Table 5-33. DST_ADDR_SHADOW Register Field Descriptions....712
Table 5-34. DST_BEG_ADDR_ACTIVE Register Field Descriptions....713



Copyright © 2024 Texas Instruments Incorporated

Table 5-35. DST_ADDR_ACTIVE Register Field Descriptions....714
Table 5-36. DMA Registers to Driverlib Functions....714
Table 6-1. Configuration Options....723
Table 6-2. Write Followed by Read - Read Occurs First....733
Table 6-3. Write Followed by Read - Write Occurs First....733
Table 6-4. ADC to CLA Early Interrupt Response....736
Table 6-5. Operand Nomenclature....739
Table 6-6. INSTRUCTION dest, source1, source2 Short Description....740
Table 6-7. Addressing Modes....741
Table 6-8. Shift Field Encoding....741
Table 6-9. Operand Encoding....742
Table 6-10. Condition Field Encoding....742
Table 6-11. Pipeline Activity for MBCNDD, Branch Not Taken....760
Table 6-12. Pipeline Activity for MBCNDD, Branch Taken....760
Table 6-13. Pipeline Activity for MCCNDD, Call Not Taken....765
Table 6-14. Pipeline Activity for MCCNDD, Call Taken....766
Table 6-15. Pipeline Activity for MMOV16 MARx, MRa , #16I....806
Table 6-16. Pipeline Activity for MMOV16 MAR0/MAR1, mem16....809
Table 6-17. Pipeline Activity for MMOVI16 MAR0/MAR1, #16I....827
Table 6-18. Pipeline Activity for MRCNDD, Return Not Taken....851
Table 6-19. Pipeline Activity for MRCNDD, Return Taken....851
Table 6-20. Pipeline Activity for MSTOP....854
Table 6-21. CLA Base Address Table....870
Table 6-22. CLA_REGS Registers....871
Table 6-23. CLA_REGS Access Type Codes....871
Table 6-24. MVECT1 Register Field Descriptions....873
Table 6-25. MVECT2 Register Field Descriptions....874
Table 6-26. MVECT3 Register Field Descriptions....875
Table 6-27. MVECT4 Register Field Descriptions....876
Table 6-28. MVECT5 Register Field Descriptions....877
Table 6-29. MVECT6 Register Field Descriptions....878
Table 6-30. MVECT7 Register Field Descriptions....879
Table 6-31. MVECT8 Register Field Descriptions....880
Table 6-32. MCTL Register Field Descriptions....881
Table 6-33. MIFR Register Field Descriptions....882
Table 6-34. MIOVF Register Field Descriptions....886
Table 6-35. MIFRC Register Field Descriptions....889
Table 6-36. MICLR Register Field Descriptions....891
Table 6-37. MICLROVF Register Field Descriptions....893
Table 6-38. MIER Register Field Descriptions....895
Table 6-39. MIRUN Register Field Descriptions....898
Table 6-40. _MPC Register Field Descriptions....900
Table 6-41. _MAR0 Register Field Descriptions....901
Table 6-42. _MAR1 Register Field Descriptions....902
Table 6-43. _MSTF Register Field Descriptions....903
Table 6-44. _MR0 Register Field Descriptions....906
Table 6-45. _MR1 Register Field Descriptions....907
Table 6-46. _MR2 Register Field Descriptions....908
Table 6-47. _MR3 Register Field Descriptions....909
Table 6-48. CLA_SOFTINT_REGS Registers....910
Table 6-49. CLA_SOFTINT_REGS Access Type Codes....910
Table 6-50. SOFTINTEN Register Field Descriptions....911
Table 6-51. SOFTINTFRC Register Field Descriptions....913
Table 6-52. CLA Registers to Driverlib Functions....914
Table 7-1. IPC Message RAM Read/Write Access....918
Table 7-2. IPC Command Registers....918
Table 7-3. IPC Base Addresses....920
Table 7-4. IPC_REGS_CPU1 Registers....921
Table 7-5. IPC_REGS_CPU1 Access Type Codes....921
Table 7-6. IPCACK Register Field Descriptions....922
Table 7-7. IPCSTS Register Field Descriptions....925



Copyright © 2024 Texas Instruments Incorporated

Table 7-8. IPCSET Register Field Descriptions....929
Table 7-9. IPCCLR Register Field Descriptions....932
Table 7-10. IPCFLG Register Field Descriptions....937
Table 7-11. IPCCOUNTERL Register Field Descriptions....941
Table 7-12. IPCCOUNTERH Register Field Descriptions....942
Table 7-13. IPCSENDCOM Register Field Descriptions....943
Table 7-14. IPCSENDADDR Register Field Descriptions....944
Table 7-15. IPCSENDDATA Register Field Descriptions....945
Table 7-16. IPCREMOTEREPLY Register Field Descriptions....946
Table 7-17. IPCRECVCOM Register Field Descriptions....947
Table 7-18. IPCRECVADDR Register Field Descriptions....948
Table 7-19. IPCRECVDATA Register Field Descriptions....949
Table 7-20. IPCLOCALREPLY Register Field Descriptions....950
Table 7-21. IPCBOOTSTS Register Field Descriptions....951
Table 7-22. IPCBOOTMODE Register Field Descriptions....952
Table 7-23. IPC_REGS_CPU2 Registers....953
Table 7-24. IPC_REGS_CPU2 Access Type Codes....953
Table 7-25. IPCACK Register Field Descriptions....954
Table 7-26. IPCSTS Register Field Descriptions....957
Table 7-27. IPCSET Register Field Descriptions....961
Table 7-28. IPCCLR Register Field Descriptions....964
Table 7-29. IPCFLG Register Field Descriptions....969
Table 7-30. IPCCOUNTERL Register Field Descriptions....973
Table 7-31. IPCCOUNTERH Register Field Descriptions....974
Table 7-32. IPCRECVCOM Register Field Descriptions....975
Table 7-33. IPCRECVADDR Register Field Descriptions....976
Table 7-34. IPCRECVDATA Register Field Descriptions....977
Table 7-35. IPCLOCALREPLY Register Field Descriptions....978
Table 7-36. IPCSENDCOM Register Field Descriptions....979
Table 7-37. IPCSENDADDR Register Field Descriptions....980
Table 7-38. IPCSENDDATA Register Field Descriptions....981
Table 7-39. IPCREMOTEREPLY Register Field Descriptions....982
Table 7-40. IPCBOOTSTS Register Field Descriptions....983
Table 7-41. IPCBOOTMODE Register Field Descriptions....984
Table 7-42. IPC Registers to Driverlib Functions....984
Table 8-1. Sampling Period....992
Table 8-2. Sampling Frequency....992
Table 8-3. Case 1: Three-Sample Sampling-Window Width....993
Table 8-4. Case 2: Six-Sample Sampling-Window Width....993
Table 8-5. USB I/O Signal Muxing....995
Table 8-6. GPIO Configuration for High-Speed SPI....995
Table 8-7. GPIO Muxed Pins....996
Table 8-8. GPIO and Peripheral Muxing....1002
Table 8-9. Peripheral Muxing (Multiple Pins Assigned)....1003
Table 8-10. Specific versus Generic Terminology for Registers....1006
Table 8-11. GPIO Base Address Table....1006
Table 8-12. GPIO_CTRL_REGS Registers....1007
Table 8-13. GPIO_CTRL_REGS Access Type Codes....1009
Table 8-14. GPACTRL Register Field Descriptions....1011
Table 8-15. GPAQSEL1 Register Field Descriptions....1012
Table 8-16. GPAQSEL2 Register Field Descriptions....1014
Table 8-17. GPAMUX1 Register Field Descriptions....1016
Table 8-18. GPAMUX2 Register Field Descriptions....1018
Table 8-19. GPADIR Register Field Descriptions....1020
Table 8-20. GPAPUD Register Field Descriptions....1022
Table 8-21. GPAINV Register Field Descriptions....1024
Table 8-22. GPAODR Register Field Descriptions....1026
Table 8-23. GPAGMUX1 Register Field Descriptions....1028
Table 8-24. GPAGMUX2 Register Field Descriptions....1029
Table 8-25. GPACSEL1 Register Field Descriptions....1030
Table 8-26. GPACSEL2 Register Field Descriptions....1031



Copyright © 2024 Texas Instruments Incorporated

Table 8-27. GPACSEL3 Register Field Descriptions....1032
Table 8-28. GPACSEL4 Register Field Descriptions....1033
Table 8-29. GPALOCK Register Field Descriptions....1034
Table 8-30. GPACR Register Field Descriptions....1036
Table 8-31. GPBCTRL Register Field Descriptions....1038
Table 8-32. GPBQSEL1 Register Field Descriptions....1039
Table 8-33. GPBQSEL2 Register Field Descriptions....1041
Table 8-34. GPBMUX1 Register Field Descriptions....1043
Table 8-35. GPBMUX2 Register Field Descriptions....1045
Table 8-36. GPBDIR Register Field Descriptions....1047
Table 8-37. GPBPUD Register Field Descriptions....1049
Table 8-38. GPBINV Register Field Descriptions....1051
Table 8-39. GPBODR Register Field Descriptions....1053
Table 8-40. GPBAMSEL Register Field Descriptions....1055
Table 8-41. GPBGMUX1 Register Field Descriptions....1057
Table 8-42. GPBGMUX2 Register Field Descriptions....1058
Table 8-43. GPBCSEL1 Register Field Descriptions....1059
Table 8-44. GPBCSEL2 Register Field Descriptions....1060
Table 8-45. GPBCSEL3 Register Field Descriptions....1061
Table 8-46. GPBCSEL4 Register Field Descriptions....1062
Table 8-47. GPBLOCK Register Field Descriptions....1063
Table 8-48. GPBCR Register Field Descriptions....1065
Table 8-49. GPCCTRL Register Field Descriptions....1067
Table 8-50. GPCQSEL1 Register Field Descriptions....1068
Table 8-51. GPCQSEL2 Register Field Descriptions....1070
Table 8-52. GPCMUX1 Register Field Descriptions....1072
Table 8-53. GPCMUX2 Register Field Descriptions....1074
Table 8-54. GPCDIR Register Field Descriptions....1076
Table 8-55. GPCPUD Register Field Descriptions....1078
Table 8-56. GPCINV Register Field Descriptions....1080
Table 8-57. GPCODR Register Field Descriptions....1082
Table 8-58. GPCGMUX1 Register Field Descriptions....1084
Table 8-59. GPCGMUX2 Register Field Descriptions....1085
Table 8-60. GPCCSEL1 Register Field Descriptions....1086
Table 8-61. GPCCSEL2 Register Field Descriptions....1087
Table 8-62. GPCCSEL3 Register Field Descriptions....1088
Table 8-63. GPCCSEL4 Register Field Descriptions....1089
Table 8-64. GPCLOCK Register Field Descriptions....1090
Table 8-65. GPCCR Register Field Descriptions....1092
Table 8-66. GPDCTRL Register Field Descriptions....1094
Table 8-67. GPDQSEL1 Register Field Descriptions....1095
Table 8-68. GPDQSEL2 Register Field Descriptions....1097
Table 8-69. GPDMUX1 Register Field Descriptions....1099
Table 8-70. GPDMUX2 Register Field Descriptions....1101
Table 8-71. GPDDIR Register Field Descriptions....1103
Table 8-72. GPDPUD Register Field Descriptions....1105
Table 8-73. GPDINV Register Field Descriptions....1107
Table 8-74. GPDODR Register Field Descriptions....1109
Table 8-75. GPDGMUX1 Register Field Descriptions....1111
Table 8-76. GPDGMUX2 Register Field Descriptions....1113
Table 8-77. GPDCSEL1 Register Field Descriptions....1115
Table 8-78. GPDCSEL2 Register Field Descriptions....1116
Table 8-79. GPDCSEL3 Register Field Descriptions....1117
Table 8-80. GPDCSEL4 Register Field Descriptions....1118
Table 8-81. GPDLOCK Register Field Descriptions....1119
Table 8-82. GPDCR Register Field Descriptions....1121
Table 8-83. GPECTRL Register Field Descriptions....1123
Table 8-84. GPEQSEL1 Register Field Descriptions....1124
Table 8-85. GPEQSEL2 Register Field Descriptions....1126
Table 8-86. GPEMUX1 Register Field Descriptions....1128
Table 8-87. GPEMUX2 Register Field Descriptions....1130



Copyright © 2024 Texas Instruments Incorporated

Table 8-88. GPEDIR Register Field Descriptions....1132
Table 8-89. GPEPUD Register Field Descriptions....1134
Table 8-90. GPEINV Register Field Descriptions....1136
Table 8-91. GPEODR Register Field Descriptions....1138
Table 8-92. GPEGMUX1 Register Field Descriptions....1140
Table 8-93. GPEGMUX2 Register Field Descriptions....1142
Table 8-94. GPECSEL1 Register Field Descriptions....1144
Table 8-95. GPECSEL2 Register Field Descriptions....1145
Table 8-96. GPECSEL3 Register Field Descriptions....1146
Table 8-97. GPECSEL4 Register Field Descriptions....1147
Table 8-98. GPELOCK Register Field Descriptions....1148
Table 8-99. GPECR Register Field Descriptions....1150
Table 8-100. GPFCTRL Register Field Descriptions....1152
Table 8-101. GPFQSEL1 Register Field Descriptions....1153
Table 8-102. GPFMUX1 Register Field Descriptions....1155
Table 8-103. GPFDIR Register Field Descriptions....1157
Table 8-104. GPFPUD Register Field Descriptions....1159
Table 8-105. GPFINV Register Field Descriptions....1161
Table 8-106. GPFODR Register Field Descriptions....1163
Table 8-107. GPFGMUX1 Register Field Descriptions....1165
Table 8-108. GPFCSEL1 Register Field Descriptions....1167
Table 8-109. GPFCSEL2 Register Field Descriptions....1168
Table 8-110. GPFLOCK Register Field Descriptions....1169
Table 8-111. GPFCR Register Field Descriptions....1171
Table 8-112. GPIO_DATA_REGS Registers....1173
Table 8-113. GPIO_DATA_REGS Access Type Codes....1173
Table 8-114. GPADAT Register Field Descriptions....1175
Table 8-115. GPASET Register Field Descriptions....1177
Table 8-116. GPACLEAR Register Field Descriptions....1179
Table 8-117. GPATOGGLE Register Field Descriptions....1181
Table 8-118. GPBDAT Register Field Descriptions....1183
Table 8-119. GPBSET Register Field Descriptions....1185
Table 8-120. GPBCLEAR Register Field Descriptions....1187
Table 8-121. GPBTOGGLE Register Field Descriptions....1189
Table 8-122. GPCDAT Register Field Descriptions....1191
Table 8-123. GPCSET Register Field Descriptions....1193
Table 8-124. GPCCLEAR Register Field Descriptions....1195
Table 8-125. GPCTOGGLE Register Field Descriptions....1197
Table 8-126. GPDDAT Register Field Descriptions....1199
Table 8-127. GPDSET Register Field Descriptions....1201
Table 8-128. GPDCLEAR Register Field Descriptions....1203
Table 8-129. GPDTOGGLE Register Field Descriptions....1205
Table 8-130. GPEDAT Register Field Descriptions....1207
Table 8-131. GPESET Register Field Descriptions....1209
Table 8-132. GPECLEAR Register Field Descriptions....1211
Table 8-133. GPETOGGLE Register Field Descriptions....1213
Table 8-134. GPFDAT Register Field Descriptions....1215
Table 8-135. GPFSET Register Field Descriptions....1217
Table 8-136. GPFCLEAR Register Field Descriptions....1219
Table 8-137. GPFTOGGLE Register Field Descriptions....1221
Table 8-138. GPIO Registers to Driverlib Functions....1222
Table 9-1. Input X-BAR Destinations....1232
Table 9-2. EPWM X-BAR Mux Configuration Table....1234
Table 9-3. CLB X-BAR Mux Configuration Table....1237
Table 9-4. Output X-BAR Mux Configuration Table....1239
Table 9-5. XBAR Base Address Table....1241
Table 9-6. INPUT_XBAR_REGS Registers....1242
Table 9-7. INPUT_XBAR_REGS Access Type Codes....1242
Table 9-8. INPUT1SELECT Register Field Descriptions....1243
Table 9-9. INPUT2SELECT Register Field Descriptions....1244
Table 9-10. INPUT3SELECT Register Field Descriptions....1245



Copyright © 2024 Texas Instruments Incorporated

Table 9-11. INPUT4SELECT Register Field Descriptions....1246
Table 9-12. INPUT5SELECT Register Field Descriptions....1247
Table 9-13. INPUT6SELECT Register Field Descriptions....1248
Table 9-14. INPUT7SELECT Register Field Descriptions....1249
Table 9-15. INPUT8SELECT Register Field Descriptions....1250
Table 9-16. INPUT9SELECT Register Field Descriptions....1251
Table 9-17. INPUT10SELECT Register Field Descriptions....1252
Table 9-18. INPUT11SELECT Register Field Descriptions....1253
Table 9-19. INPUT12SELECT Register Field Descriptions....1254
Table 9-20. INPUT13SELECT Register Field Descriptions....1255
Table 9-21. INPUT14SELECT Register Field Descriptions....1256
Table 9-22. INPUTSELECTLOCK Register Field Descriptions....1257
Table 9-23. XBAR_REGS Registers....1259
Table 9-24. XBAR_REGS Access Type Codes....1259
Table 9-25. XBARFLG1 Register Field Descriptions....1260
Table 9-26. XBARFLG2 Register Field Descriptions....1262
Table 9-27. XBARFLG3 Register Field Descriptions....1264
Table 9-28. XBARCLR1 Register Field Descriptions....1266
Table 9-29. XBARCLR2 Register Field Descriptions....1268
Table 9-30. XBARCLR3 Register Field Descriptions....1270
Table 9-31. EPWM_XBAR_REGS Registers....1272
Table 9-32. EPWM_XBAR_REGS Access Type Codes....1272
Table 9-33. TRIP4MUX0TO15CFG Register Field Descriptions....1274
Table 9-34. TRIP4MUX16TO31CFG Register Field Descriptions....1277
Table 9-35. TRIP5MUX0TO15CFG Register Field Descriptions....1280
Table 9-36. TRIP5MUX16TO31CFG Register Field Descriptions....1283
Table 9-37. TRIP7MUX0TO15CFG Register Field Descriptions....1286
Table 9-38. TRIP7MUX16TO31CFG Register Field Descriptions....1289
Table 9-39. TRIP8MUX0TO15CFG Register Field Descriptions....1292
Table 9-40. TRIP8MUX16TO31CFG Register Field Descriptions....1295
Table 9-41. TRIP9MUX0TO15CFG Register Field Descriptions....1298
Table 9-42. TRIP9MUX16TO31CFG Register Field Descriptions....1301
Table 9-43. TRIP10MUX0TO15CFG Register Field Descriptions....1304
Table 9-44. TRIP10MUX16TO31CFG Register Field Descriptions....1307
Table 9-45. TRIP11MUX0TO15CFG Register Field Descriptions....1310
Table 9-46. TRIP11MUX16TO31CFG Register Field Descriptions....1313
Table 9-47. TRIP12MUX0TO15CFG Register Field Descriptions....1316
Table 9-48. TRIP12MUX16TO31CFG Register Field Descriptions....1319
Table 9-49. TRIP4MUXENABLE Register Field Descriptions....1322
Table 9-50. TRIP5MUXENABLE Register Field Descriptions....1327
Table 9-51. TRIP7MUXENABLE Register Field Descriptions....1332
Table 9-52. TRIP8MUXENABLE Register Field Descriptions....1337
Table 9-53. TRIP9MUXENABLE Register Field Descriptions....1342
Table 9-54. TRIP10MUXENABLE Register Field Descriptions....1347
Table 9-55. TRIP11MUXENABLE Register Field Descriptions....1352
Table 9-56. TRIP12MUXENABLE Register Field Descriptions....1357
Table 9-57. TRIPOUTINV Register Field Descriptions....1362
Table 9-58. TRIPLOCK Register Field Descriptions....1364
Table 9-59. CLB_XBAR_REGS Registers....1365
Table 9-60. CLB_XBAR_REGS Access Type Codes....1365
Table 9-61. AUXSIG0MUX0TO15CFG Register Field Descriptions....1367
Table 9-62. AUXSIG0MUX16TO31CFG Register Field Descriptions....1370
Table 9-63. AUXSIG1MUX0TO15CFG Register Field Descriptions....1373
Table 9-64. AUXSIG1MUX16TO31CFG Register Field Descriptions....1376
Table 9-65. AUXSIG2MUX0TO15CFG Register Field Descriptions....1379
Table 9-66. AUXSIG2MUX16TO31CFG Register Field Descriptions....1382
Table 9-67. AUXSIG3MUX0TO15CFG Register Field Descriptions....1385
Table 9-68. AUXSIG3MUX16TO31CFG Register Field Descriptions....1388
Table 9-69. AUXSIG4MUX0TO15CFG Register Field Descriptions....1391
Table 9-70. AUXSIG4MUX16TO31CFG Register Field Descriptions....1394
Table 9-71. AUXSIG5MUX0TO15CFG Register Field Descriptions....1397



Copyright © 2024 Texas Instruments Incorporated

Table 9-72. AUXSIG5MUX16TO31CFG Register Field Descriptions....1400
Table 9-73. AUXSIG6MUX0TO15CFG Register Field Descriptions....1403
Table 9-74. AUXSIG6MUX16TO31CFG Register Field Descriptions....1406
Table 9-75. AUXSIG7MUX0TO15CFG Register Field Descriptions....1409
Table 9-76. AUXSIG7MUX16TO31CFG Register Field Descriptions....1412
Table 9-77. AUXSIG0MUXENABLE Register Field Descriptions....1415
Table 9-78. AUXSIG1MUXENABLE Register Field Descriptions....1420
Table 9-79. AUXSIG2MUXENABLE Register Field Descriptions....1425
Table 9-80. AUXSIG3MUXENABLE Register Field Descriptions....1430
Table 9-81. AUXSIG4MUXENABLE Register Field Descriptions....1435
Table 9-82. AUXSIG5MUXENABLE Register Field Descriptions....1440
Table 9-83. AUXSIG6MUXENABLE Register Field Descriptions....1445
Table 9-84. AUXSIG7MUXENABLE Register Field Descriptions....1450
Table 9-85. AUXSIGOUTINV Register Field Descriptions....1455
Table 9-86. AUXSIGLOCK Register Field Descriptions....1457
Table 9-87. OUTPUT_XBAR_REGS Registers....1458
Table 9-88. OUTPUT_XBAR_REGS Access Type Codes....1458
Table 9-89. OUTPUT1MUX0TO15CFG Register Field Descriptions....1460
Table 9-90. OUTPUT1MUX16TO31CFG Register Field Descriptions....1463
Table 9-91. OUTPUT2MUX0TO15CFG Register Field Descriptions....1466
Table 9-92. OUTPUT2MUX16TO31CFG Register Field Descriptions....1469
Table 9-93. OUTPUT3MUX0TO15CFG Register Field Descriptions....1472
Table 9-94. OUTPUT3MUX16TO31CFG Register Field Descriptions....1475
Table 9-95. OUTPUT4MUX0TO15CFG Register Field Descriptions....1478
Table 9-96. OUTPUT4MUX16TO31CFG Register Field Descriptions....1481
Table 9-97. OUTPUT5MUX0TO15CFG Register Field Descriptions....1484
Table 9-98. OUTPUT5MUX16TO31CFG Register Field Descriptions....1487
Table 9-99. OUTPUT6MUX0TO15CFG Register Field Descriptions....1490
Table 9-100. OUTPUT6MUX16TO31CFG Register Field Descriptions....1493
Table 9-101. OUTPUT7MUX0TO15CFG Register Field Descriptions....1496
Table 9-102. OUTPUT7MUX16TO31CFG Register Field Descriptions....1499
Table 9-103. OUTPUT8MUX0TO15CFG Register Field Descriptions....1502
Table 9-104. OUTPUT8MUX16TO31CFG Register Field Descriptions....1505
Table 9-105. OUTPUT1MUXENABLE Register Field Descriptions....1508
Table 9-106. OUTPUT2MUXENABLE Register Field Descriptions....1513
Table 9-107. OUTPUT3MUXENABLE Register Field Descriptions....1518
Table 9-108. OUTPUT4MUXENABLE Register Field Descriptions....1523
Table 9-109. OUTPUT5MUXENABLE Register Field Descriptions....1528
Table 9-110. OUTPUT6MUXENABLE Register Field Descriptions....1533
Table 9-111. OUTPUT7MUXENABLE Register Field Descriptions....1538
Table 9-112. OUTPUT8MUXENABLE Register Field Descriptions....1543
Table 9-113. OUTPUTLATCH Register Field Descriptions....1548
Table 9-114. OUTPUTLATCHCLR Register Field Descriptions....1550
Table 9-115. OUTPUTLATCHFRC Register Field Descriptions....1552
Table 9-116. OUTPUTLATCHENABLE Register Field Descriptions....1554
Table 9-117. OUTPUTINV Register Field Descriptions....1556
Table 9-118. OUTPUTLOCK Register Field Descriptions....1558
Table 9-119. INPUTXBAR Registers to Driverlib Functions....1559
Table 9-120. XBAR Registers to Driverlib Functions....1559
Table 9-121. EPWMXBAR Registers to Driverlib Functions....1560
Table 9-122. CLBXBAR Registers to Driverlib Functions....1561
Table 9-123. OUTPUTXBAR Registers to Driverlib Functions....1562
Table 10-1. Analog Signal Descriptions....1570
Table 10-2. Reference Summary....1570
Table 10-3. Analog Subsystem Base Address Table....1571
Table 10-4. ANALOG_SUBSYS_REGS Registers....1572
Table 10-5. ANALOG_SUBSYS_REGS Access Type Codes....1572
Table 10-6. INTOSC1TRIM Register Field Descriptions....1573
Table 10-7. INTOSC2TRIM Register Field Descriptions....1574
Table 10-8. TSNSCTL Register Field Descriptions....1575
Table 10-9. LOCK Register Field Descriptions....1576



Copyright © 2024 Texas Instruments Incorporated

Table 10-10. ANAREFTRIMA Register Field Descriptions ... 1578
Table 10-11. ANAREFTRIMB Register Field Descriptions ... 1579
Table 10-12. ANAREFTRIMC Register Field Descriptions ... 1580
Table 10-13. ANAREFTRIMD Register Field Descriptions ... 1581
Table 11-1. ADC Options and Configuration Levels ... 1586
Table 11-2. Analog to 12-bit Digital Formulas ... 1588
Table 11-3. Analog to 16-bit Digital Formulas ... 1588
Table 11-4. 12-Bit Digital-to-Analog Formulas ... 1589
Table 11-5. 16-Bit Digital-to-Analog Formulas ... 1589
Table 11-6. Channel Selection of Input Pins ... 1593
Table 11-7. Example Requirements for Multiple Signal Sampling ... 1595
Table 11-8. Example Connections for Multiple Signal Sampling ... 1595
Table 11-9. DETECTCFG Settings ... 1607
Table 11-10. ADC Calibration Functions in OTP ... 1611
Table 11-11. ADC Timing Parameter Descriptions ... 1612
Table 11-12. ADC Timings in 12-bit Mode ... 1617
Table 11-13. ADC Timings in 16-bit Mode ... 1617
Table 11-14. ADC Base Address Table ... 1630
Table 11-15. ADC_RESULT_REGS Registers ... 1631
Table 11-16. ADC_RESULT_REGS Access Type Codes ... 1631
Table 11-17. ADCRESULT0 Register Field Descriptions ... 1632
Table 11-18. ADCRESULT1 Register Field Descriptions ... 1633
Table 11-19. ADCRESULT2 Register Field Descriptions ... 1634
Table 11-20. ADCRESULT3 Register Field Descriptions ... 1635
Table 11-21. ADCRESULT4 Register Field Descriptions ... 1636
Table 11-22. ADCRESULT5 Register Field Descriptions ... 1637
Table 11-23. ADCRESULT6 Register Field Descriptions ... 1638
Table 11-24. ADCRESULT7 Register Field Descriptions ... 1639
Table 11-25. ADCRESULT8 Register Field Descriptions ... 1640
Table 11-26. ADCRESULT9 Register Field Descriptions ... 1641
Table 11-27. ADCRESULT10 Register Field Descriptions ... 1642
Table 11-28. ADCRESULT11 Register Field Descriptions ... 1643
Table 11-29. ADCRESULT12 Register Field Descriptions ... 1644
Table 11-30. ADCRESULT13 Register Field Descriptions ... 1645
Table 11-31. ADCRESULT14 Register Field Descriptions ... 1646
Table 11-32. ADCRESULT15 Register Field Descriptions ... 1647
Table 11-33. ADCPPB1RESULT Register Field Descriptions ... 1648
Table 11-34. ADCPPB2RESULT Register Field Descriptions ... 1649
Table 11-35. ADCPPB3RESULT Register Field Descriptions ... 1650
Table 11-36. ADCPPB4RESULT Register Field Descriptions ... 1651
Table 11-37. ADC_REGS Registers ... 1652
Table 11-38. ADC_REGS Access Type Codes ... 1653
Table 11-39. ADCCTL1 Register Field Descriptions ... 1655
Table 11-40. ADCCTL2 Register Field Descriptions ... 1657
Table 11-41. ADCBURSTCTL Register Field Descriptions ... 1658
Table 11-42. ADCINTFLG Register Field Descriptions ... 1660
Table 11-43. ADCINTFLGCLR Register Field Descriptions ... 1662
Table 11-44. ADCINTOVF Register Field Descriptions ... 1663
Table 11-45. ADCINTOVFCLR Register Field Descriptions ... 1664
Table 11-46. ADCINTSEL1N2 Register Field Descriptions ... 1665
Table 11-47. ADCINTSEL3N4 Register Field Descriptions ... 1667
Table 11-48. ADCSOCPRICTL Register Field Descriptions ... 1669
Table 11-49. ADCINTSOCSEL1 Register Field Descriptions ... 1671
Table 11-50. ADCINTSOCSEL2 Register Field Descriptions ... 1673
Table 11-51. ADCSOCFLG1 Register Field Descriptions ... 1675
Table 11-52. ADCSOCFRC1 Register Field Descriptions ... 1679
Table 11-53. ADCSOCOVF1 Register Field Descriptions ... 1684
Table 11-54. ADCSOCOVFCLR1 Register Field Descriptions ... 1687
Table 11-55. ADCSOC0CTL Register Field Descriptions ... 1690
Table 11-56. ADCSOC1CTL Register Field Descriptions ... 1693
Table 11-57. ADCSOC2CTL Register Field Descriptions ... 1696



Copyright © 2024 Texas Instruments Incorporated

Table 11-58. ADCSOC3CTL Register Field Descriptions....1699
Table 11-59. ADCSOC4CTL Register Field Descriptions....1702
Table 11-60. ADCSOC5CTL Register Field Descriptions....1705
Table 11-61. ADCSOC6CTL Register Field Descriptions....1708
Table 11-62. ADCSOC7CTL Register Field Descriptions....1711
Table 11-63. ADCSOC8CTL Register Field Descriptions....1714
Table 11-64. ADCSOC9CTL Register Field Descriptions....1717
Table 11-65. ADCSOC10CTL Register Field Descriptions....1720
Table 11-66. ADCSOC11CTL Register Field Descriptions....1723
Table 11-67. ADCSOC12CTL Register Field Descriptions....1726
Table 11-68. ADCSOC13CTL Register Field Descriptions....1729
Table 11-69. ADCSOC14CTL Register Field Descriptions....1732
Table 11-70. ADCSOC15CTL Register Field Descriptions....1735
Table 11-71. ADCEVTSTAT Register Field Descriptions....1738
Table 11-72. ADCEVTCLR Register Field Descriptions....1741
Table 11-73. ADCEVTSEL Register Field Descriptions....1743
Table 11-74. ADCEVTINTSEL Register Field Descriptions....1745
Table 11-75. ADCOSDETECT Register Field Descriptions....1747
Table 11-76. ADCCOUNTER Register Field Descriptions....1748
Table 11-77. ADCREV Register Field Descriptions....1749
Table 11-78. ADCOFFTRIM Register Field Descriptions....1750
Table 11-79. ADCPPB1CONFIG Register Field Descriptions....1751
Table 11-80. ADCPPB1STAMP Register Field Descriptions....1753
Table 11-81. ADCPPB1OFFCAL Register Field Descriptions....1754
Table 11-82. ADCPPB1OFFREF Register Field Descriptions....1755
Table 11-83. ADCPPB1TRIPHI Register Field Descriptions....1756
Table 11-84. ADCPPB1TRIPLO Register Field Descriptions....1757
Table 11-85. ADCPPB2CONFIG Register Field Descriptions....1758
Table 11-86. ADCPPB2STAMP Register Field Descriptions....1760
Table 11-87. ADCPPB2OFFCAL Register Field Descriptions....1761
Table 11-88. ADCPPB2OFFREF Register Field Descriptions....1762
Table 11-89. ADCPPB2TRIPHI Register Field Descriptions....1763
Table 11-90. ADCPPB2TRIPLO Register Field Descriptions....1764
Table 11-91. ADCPPB3CONFIG Register Field Descriptions....1765
Table 11-92. ADCPPB3STAMP Register Field Descriptions....1767
Table 11-93. ADCPPB3OFFCAL Register Field Descriptions....1768
Table 11-94. ADCPPB3OFFREF Register Field Descriptions....1769
Table 11-95. ADCPPB3TRIPHI Register Field Descriptions....1770
Table 11-96. ADCPPB3TRIPLO Register Field Descriptions....1771
Table 11-97. ADCPPB4CONFIG Register Field Descriptions....1772
Table 11-98. ADCPPB4STAMP Register Field Descriptions....1774
Table 11-99. ADCPPB4OFFCAL Register Field Descriptions....1775
Table 11-100. ADCPPB4OFFREF Register Field Descriptions....1776
Table 11-101. ADCPPB4TRIPHI Register Field Descriptions....1777
Table 11-102. ADCPPB4TRIPLO Register Field Descriptions....1778
Table 11-103. ADCINLTRIM1 Register Field Descriptions....1779
Table 11-104. ADCINLTRIM2 Register Field Descriptions....1780
Table 11-105. ADCINLTRIM3 Register Field Descriptions....1781
Table 11-106. ADCINLTRIM4 Register Field Descriptions....1782
Table 11-107. ADCINLTRIM5 Register Field Descriptions....1783
Table 11-108. ADCINLTRIM6 Register Field Descriptions....1784
Table 11-109. ADC Registers to Driverlib Functions....1784
Table 12-1. DAC Base Address Table....1793
Table 12-2. DAC_REGS Registers....1794
Table 12-3. DAC_REGS Access Type Codes....1794
Table 12-4. DACREV Register Field Descriptions....1795
Table 12-5. DACCTL Register Field Descriptions....1796
Table 12-6. DACVALA Register Field Descriptions....1797
Table 12-7. DACVALS Register Field Descriptions....1798
Table 12-8. DACOUTEN Register Field Descriptions....1799
Table 12-9. DACLOCK Register Field Descriptions....1800



Copyright © 2024 Texas Instruments Incorporated

Table 12-10. DACTRIM Register Field Descriptions....1801
Table 12-11. DAC Registers to Driverlib Functions....1801
Table 13-1. CMPSS Base Address Table....1814
Table 13-2. CMPSS_REGS Registers....1815
Table 13-3. CMPSS_REGS Access Type Codes....1815
Table 13-4. COMPCTL Register Field Descriptions....1817
Table 13-5. COMPHYSCTL Register Field Descriptions....1819
Table 13-6. COMPSTS Register Field Descriptions....1820
Table 13-7. COMPSTSCLR Register Field Descriptions....1821
Table 13-8. COMPDACCTL Register Field Descriptions....1822
Table 13-9. DACHVALS Register Field Descriptions....1823
Table 13-10. DACHVALA Register Field Descriptions....1824
Table 13-11. RAMPMAXREFA Register Field Descriptions....1825
Table 13-12. RAMPMAXREFS Register Field Descriptions....1826
Table 13-13. RAMPDECVALA Register Field Descriptions....1827
Table 13-14. RAMPDECVALS Register Field Descriptions....1828
Table 13-15. RAMPSTS Register Field Descriptions....1829
Table 13-16. DACLVALS Register Field Descriptions....1830
Table 13-17. DACLVALA Register Field Descriptions....1831
Table 13-18. RAMPDLYA Register Field Descriptions....1832
Table 13-19. RAMPDLYS Register Field Descriptions....1833
Table 13-20. CTRIPLFILCTL Register Field Descriptions....1834
Table 13-21. CTRIPLFILCLKCTL Register Field Descriptions....1835
Table 13-22. CTRIPHFILCTL Register Field Descriptions....1836
Table 13-23. CTRIPHFILCLKCTL Register Field Descriptions....1837
Table 13-24. COMPLOCK Register Field Descriptions....1838
Table 13-25. CMPSS Registers to Driverlib Functions....1838
Table 14-1. Modulator Clock Modes....1847
Table 14-2. Order of Sinc Filter....1850
Table 14-3. Peak Data Values for Different DOSR/Filter Combinations....1851
Table 14-4. Shift Control Bit Configuration Settings....1852
Table 14-5. Number of Incorrect Samples Tabulated....1853
Table 14-6. Peak Data Values for Different OSR/Filter Combinations....1854
Table 14-7. General Registers....1860
Table 14-8. Filter 1 Registers....1860
Table 14-9. Filter 2 Registers....1860
Table 14-10. Filter 3 Registers....1861
Table 14-11. Filter 4 Registers....1861
Table 14-12. SDFM Base Address Table....1862
Table 14-13. SDFM_REGS Registers....1863
Table 14-14. SDFM_REGS Access Type Codes....1863
Table 14-15. SDIFLG Register Field Descriptions....1865
Table 14-16. SDIFLGCLR Register Field Descriptions....1867
Table 14-17. SDCTL Register Field Descriptions....1869
Table 14-18. SDMFILEN Register Field Descriptions....1870
Table 14-19. SDCTLPARM1 Register Field Descriptions....1871
Table 14-20. SDDFPARM1 Register Field Descriptions....1872
Table 14-21. SDDPARM1 Register Field Descriptions....1873
Table 14-22. SDCMPH1 Register Field Descriptions....1874
Table 14-23. SDCMPL1 Register Field Descriptions....1875
Table 14-24. SDCPARM1 Register Field Descriptions....1876
Table 14-25. SDDATA1 Register Field Descriptions....1877
Table 14-26. SDCTLPARM2 Register Field Descriptions....1878
Table 14-27. SDDFPARM2 Register Field Descriptions....1879
Table 14-28. SDDPARM2 Register Field Descriptions....1880
Table 14-29. SDCMPH2 Register Field Descriptions....1881
Table 14-30. SDCMPL2 Register Field Descriptions....1882
Table 14-31. SDCPARM2 Register Field Descriptions....1883
Table 14-32. SDDATA2 Register Field Descriptions....1884
Table 14-33. SDCTLPARM3 Register Field Descriptions....1885
Table 14-34. SDDFPARM3 Register Field Descriptions....1886



Copyright © 2024 Texas Instruments Incorporated

Table 14-35. SDDPARM3 Register Field Descriptions ... 1887
Table 14-36. SDCMPH3 Register Field Descriptions ... 1888
Table 14-37. SDCMPL3 Register Field Descriptions ... 1889
Table 14-38. SDCPARM3 Register Field Descriptions ... 1890
Table 14-39. SDDATA3 Register Field Descriptions ... 1891
Table 14-40. SDCTLPARM4 Register Field Descriptions ... 1892
Table 14-41. SDDFPARM4 Register Field Descriptions ... 1893
Table 14-42. SDDPARM4 Register Field Descriptions ... 1894
Table 14-43. SDCMPH4 Register Field Descriptions ... 1895
Table 14-44. SDCMPL4 Register Field Descriptions ... 1896
Table 14-45. SDCPARM4 Register Field Descriptions ... 1897
Table 14-46. SDDATA4 Register Field Descriptions ... 1898
Table 14-47. SDFM Registers to Driverlib Functions ... 1899
Table 15-1. Submodule Configuration Parameters ... 1909
Table 15-2. Key Time-Base Signals ... 1913
Table 15-3. Action-Qualifier Submodule Possible Input Events ... 1931
Table 15-4. Action-Qualifier Event Priority for Up-Down-Count Mode ... 1933
Table 15-5. Action-Qualifier Event Priority for Up-Count Mode ... 1933
Table 15-6. Action-Qualifier Event Priority for Down-Count Mode ... 1933
Table 15-7. Behavior if CMPA/CMPB is Greater than the Period ... 1934
Table 15-8. Classical Dead-Band Operating Modes ... 1947
Table 15-9. Additional Dead-Band Operating Modes ... 1947
Table 15-10. Dead-Band Delay Values in µs as a Function of DBFED and DBRED ... 1949
Table 15-11. Possible Pulse Width Values for EPWMCLK = 80MHz ... 1952
Table 15-12. Possible Actions On a Trip Event ... 1956
Table 15-13. Resolution for PWM and HRPWM ... 1995
Table 15-14. Relationship Between MEP Steps, PWM Frequency, and Resolution ... 2001
Table 15-15. CMPA versus Duty (left), and [CMPA:CMPAHR] versus Duty (right) ... 2002
Table 15-16. Duty Cycle Range Limitation for Three EPWMCLK/TBCLK Cycles ... 2005
Table 15-17. SFO Library Features ... 2017
Table 15-18. Factor Values ... 2018
Table 15-19. ePWM Base Address Table ... 2020
Table 15-20. EPWM_REGS Registers ... 2021
Table 15-21. EPWM_REGS Access Type Codes ... 2023
Table 15-22. TBCTL Register Field Descriptions ... 2024
Table 15-23. TBCTL2 Register Field Descriptions ... 2026
Table 15-24. TBCTR Register Field Descriptions ... 2027
Table 15-25. TBSTS Register Field Descriptions ... 2028
Table 15-26. CMPCTL Register Field Descriptions ... 2029
Table 15-27. CMPCTL2 Register Field Descriptions ... 2031
Table 15-28. DBCTL Register Field Descriptions ... 2033
Table 15-29. DBCTL2 Register Field Descriptions ... 2036
Table 15-30. AQCTL Register Field Descriptions ... 2037
Table 15-31. AQTSRCSEL Register Field Descriptions ... 2039
Table 15-32. PCCTL Register Field Descriptions ... 2040
Table 15-33. VCAPCTL Register Field Descriptions ... 2042
Table 15-34. VCNTCFG Register Field Descriptions ... 2044
Table 15-35. HRCNFG Register Field Descriptions ... 2046
Table 15-36. HRPWR Register Field Descriptions ... 2048
Table 15-37. HRMSTEP Register Field Descriptions ... 2049
Table 15-38. HRCNFG2 Register Field Descriptions ... 2050
Table 15-39. HRPCTL Register Field Descriptions ... 2051
Table 15-40. TRREM Register Field Descriptions ... 2053
Table 15-41. GLDCTL Register Field Descriptions ... 2054
Table 15-42. GLDCFG Register Field Descriptions ... 2056
Table 15-43. EPWMXLINK Register Field Descriptions ... 2058
Table 15-44. AQCTLA Register Field Descriptions ... 2060
Table 15-45. AQCTLA2 Register Field Descriptions ... 2062
Table 15-46. AQCTLB Register Field Descriptions ... 2063
Table 15-47. AQCTLB2 Register Field Descriptions ... 2065
Table 15-48. AQSFRC Register Field Descriptions ... 2066



Copyright © 2024 Texas Instruments Incorporated

Table 15-49. AQCSFRC Register Field Descriptions ..... 2067
Table 15-50. DBREDHR Register Field Descriptions ..... 2068
Table 15-51. DBRED Register Field Descriptions ..... 2069
Table 15-52. DBFEDHR Register Field Descriptions ..... 2070
Table 15-53. DBFED Register Field Descriptions ..... 2071
Table 15-54. TBPHS Register Field Descriptions ..... 2072
Table 15-55. TBPRDHR Register Field Descriptions ..... 2073
Table 15-56. TBPRD Register Field Descriptions ..... 2074
Table 15-57. CMPA Register Field Descriptions ..... 2075
Table 15-58. CMPB Register Field Descriptions ..... 2076
Table 15-59. CMPC Register Field Descriptions ..... 2077
Table 15-60. CMPD Register Field Descriptions ..... 2078
Table 15-61. GLDCTL2 Register Field Descriptions ..... 2079
Table 15-62. SWVDELVAL Register Field Descriptions ..... 2080
Table 15-63. TZSEL Register Field Descriptions ..... 2081
Table 15-64. TZDCSEL Register Field Descriptions ..... 2083
Table 15-65. TZCTL Register Field Descriptions ..... 2084
Table 15-66. TZCTL2 Register Field Descriptions ..... 2086
Table 15-67. TZCTLDCA Register Field Descriptions ..... 2088
Table 15-68. TZCTLDCB Register Field Descriptions ..... 2090
Table 15-69. TZEINT Register Field Descriptions ..... 2092
Table 15-70. TZFLG Register Field Descriptions ..... 2093
Table 15-71. TZCBCFLG Register Field Descriptions ..... 2095
Table 15-72. TZOSTFLG Register Field Descriptions ..... 2097
Table 15-73. TZCLR Register Field Descriptions ..... 2099
Table 15-74. TZCBCCLR Register Field Descriptions ..... 2101
Table 15-75. TZOSTCLR Register Field Descriptions ..... 2102
Table 15-76. TZFRC Register Field Descriptions ..... 2103
Table 15-77. ETSEL Register Field Descriptions ..... 2104
Table 15-78. ETPS Register Field Descriptions ..... 2107
Table 15-79. ETFLG Register Field Descriptions ..... 2110
Table 15-80. ETCLR Register Field Descriptions ..... 2111
Table 15-81. ETFRC Register Field Descriptions ..... 2112
Table 15-82. ETINTPS Register Field Descriptions ..... 2113
Table 15-83. ETSOCPS Register Field Descriptions ..... 2114
Table 15-84. ETCNTINITCTL Register Field Descriptions ..... 2116
Table 15-85. ETCNTINIT Register Field Descriptions ..... 2117
Table 15-86. DCTRIPSEL Register Field Descriptions ..... 2118
Table 15-87. DCACTL Register Field Descriptions ..... 2120
Table 15-88. DCBCTL Register Field Descriptions ..... 2121
Table 15-89. DCFCTL Register Field Descriptions ..... 2122
Table 15-90. DCCAPCTL Register Field Descriptions ..... 2124
Table 15-91. DCFOFFSET Register Field Descriptions ..... 2126
Table 15-92. DCFOFFSETCNT Register Field Descriptions ..... 2127
Table 15-93. DCFWINDOW Register Field Descriptions ..... 2128
Table 15-94. DCFWINDOWCNT Register Field Descriptions ..... 2129
Table 15-95. DCCAP Register Field Descriptions ..... 2130
Table 15-96. DCAHTRIPSEL Register Field Descriptions ..... 2131
Table 15-97. DCALTRIPSEL Register Field Descriptions ..... 2133
Table 15-98. DCBHTRIPSEL Register Field Descriptions ..... 2135
Table 15-99. DCBLTRIPSEL Register Field Descriptions ..... 2137
Table 15-100. HWVDELVAL Register Field Descriptions ..... 2139
Table 15-101. VCNTVAL Register Field Descriptions ..... 2140
Table 15-102. EPWM Registers to Driverlib Functions ..... 2141
Table 15-103. HRPWM Registers to Driverlib Functions ..... 2147
Table 16-1. eCAP Base Address Table ..... 2173
Table 16-2. ECAP_REGS Registers ..... 2174
Table 16-3. ECAP_REGS Access Type Codes ..... 2174
Table 16-4. TSCTR Register Field Descriptions ..... 2175
Table 16-5. CTRPHS Register Field Descriptions ..... 2176
Table 16-6. CAP1 Register Field Descriptions ..... 2177



Copyright © 2024 Texas Instruments Incorporated

Table 16-7. CAP2 Register Field Descriptions....2178
Table 16-8. CAP3 Register Field Descriptions....2179
Table 16-9. CAP4 Register Field Descriptions....2180
Table 16-10. ECCTL1 Register Field Descriptions....2181
Table 16-11. ECCTL2 Register Field Descriptions....2183
Table 16-12. ECEINT Register Field Descriptions....2185
Table 16-13. ECFLG Register Field Descriptions....2187
Table 16-14. ECCLR Register Field Descriptions....2188
Table 16-15. ECFRC Register Field Descriptions....2189
Table 16-16. ECAP Registers to Driverlib Functions....2189
Table 17-1. EQEP Memory Map....2198
Table 17-2. Quadrature Decoder Truth Table....2200
Table 17-3. eQEP Base Address Table....2215
Table 17-4. EQEP_REGS Registers....2216
Table 17-5. EQEP_REGS Access Type Codes....2216
Table 17-6. QPOSCNT Register Field Descriptions....2217
Table 17-7. QPOSINIT Register Field Descriptions....2218
Table 17-8. QPOSMAX Register Field Descriptions....2219
Table 17-9. QPOSCMP Register Field Descriptions....2220
Table 17-10. QPOSILAT Register Field Descriptions....2221
Table 17-11. QPOSSLAT Register Field Descriptions....2222
Table 17-12. QPOSLAT Register Field Descriptions....2223
Table 17-13. QUTMR Register Field Descriptions....2224
Table 17-14. QUPRD Register Field Descriptions....2225
Table 17-15. QWDTMR Register Field Descriptions....2226
Table 17-16. QWDPRD Register Field Descriptions....2227
Table 17-17. QDECCTL Register Field Descriptions....2228
Table 17-18. QEPCTL Register Field Descriptions....2230
Table 17-19. QCAPCTL Register Field Descriptions....2232
Table 17-20. QPOSCTL Register Field Descriptions....2233
Table 17-21. QEINT Register Field Descriptions....2234
Table 17-22. QFLG Register Field Descriptions....2236
Table 17-23. QCLR Register Field Descriptions....2238
Table 17-24. QFRC Register Field Descriptions....2240
Table 17-25. QEPSTS Register Field Descriptions....2242
Table 17-26. QCTMR Register Field Descriptions....2244
Table 17-27. QCPRD Register Field Descriptions....2245
Table 17-28. QCTMRLAT Register Field Descriptions....2246
Table 17-29. QCPRDLAT Register Field Descriptions....2247
Table 17-30. EQEP Registers to Driverlib Functions....2247
Table 18-1. SPI Module Signal Summary....2253
Table 18-2. SPI Interrupt Flag Modes....2255
Table 18-3. SPI Clocking Scheme Selection Guide....2263
Table 18-4. 4-wire versus 3-wire SPI Pin Functions....2266
Table 18-5. 3-Wire SPI Pin Configuration....2267
Table 18-6. SPI Base Address Table....2277
Table 18-7. SPI_REGS Registers....2278
Table 18-8. SPI_REGS Access Type Codes....2278
Table 18-9. SPICCR Register Field Descriptions....2279
Table 18-10. SPICTL Register Field Descriptions....2281
Table 18-11. SPISTS Register Field Descriptions....2283
Table 18-12. SPIBRR Register Field Descriptions....2285
Table 18-13. SPIRXEMU Register Field Descriptions....2286
Table 18-14. SPIRXBUF Register Field Descriptions....2287
Table 18-15. SPITXBUF Register Field Descriptions....2288
Table 18-16. SPIDAT Register Field Descriptions....2289
Table 18-17. SPIFFTX Register Field Descriptions....2290
Table 18-18. SPIFFRX Register Field Descriptions....2292
Table 18-19. SPIFFCT Register Field Descriptions....2294
Table 18-20. SPIPRI Register Field Descriptions....2295
Table 18-21. SPI Registers to Driverlib Functions....2296



Copyright © 2024 Texas Instruments Incorporated

Table 19-1. SCI Module Signal Summary....2300
Table 19-2. Programming the Data Format Using SCICCR....2303
Table 19-3. Asynchronous Baud Register Values for Common SCI Bit Rates....2312
Table 19-4. SCI Interrupt Flags....2314
Table 19-5. SCI Base Address Table....2316
Table 19-6. SCI_REGS Registers....2317
Table 19-7. SCI_REGS Access Type Codes....2317
Table 19-8. SCICCR Register Field Descriptions....2318
Table 19-9. SCICTL1 Register Field Descriptions....2320
Table 19-10. SCIHBAUD Register Field Descriptions....2322
Table 19-11. SCILBAUD Register Field Descriptions....2323
Table 19-12. SCICTL2 Register Field Descriptions....2324
Table 19-13. SCIRXST Register Field Descriptions....2326
Table 19-14. SCIRXEMU Register Field Descriptions....2329
Table 19-15. SCIRXBUF Register Field Descriptions....2330
Table 19-16. SCITXBUF Register Field Descriptions....2332
Table 19-17. SCIFFTX Register Field Descriptions....2333
Table 19-18. SCIFFRX Register Field Descriptions....2335
Table 19-19. SCIFFCT Register Field Descriptions....2337
Table 19-20. SCIPRI Register Field Descriptions....2338
Table 19-21. SCI Registers to Driverlib Functions....2338
Table 20-1. Dependency of Delay d on the Divide-Down Value IPSC....2346
Table 20-2. Operating Modes of the I2C Module....2348
Table 20-3. Master-Transmitter/Receiver Bus Activity Defined by the RM, STT, and STP Bits of I2CMDR....2348
Table 20-4. How the MST and FDF Bits of I2CMDR Affect the Role of the TRX Bit of I2CMDR....2354
Table 20-5. Ways to Generate a NACK Bit....2358
Table 20-6. Descriptions of the Basic I2C Interrupt Requests....2359
Table 20-7. I2C Base Address Table (C28)....2365
Table 20-8. I2C_REGS Registers....2366
Table 20-9. I2C_REGS Access Type Codes....2366
Table 20-10. I2COAR Register Field Descriptions....2367
Table 20-11. I2CIER Register Field Descriptions....2368
Table 20-12. I2CSTR Register Field Descriptions....2369
Table 20-13. I2CCLKL Register Field Descriptions....2373
Table 20-14. I2CCLKH Register Field Descriptions....2374
Table 20-15. I2CCNT Register Field Descriptions....2375
Table 20-16. I2CDRR Register Field Descriptions....2376
Table 20-17. I2CSAR Register Field Descriptions....2377
Table 20-18. I2CDXR Register Field Descriptions....2378
Table 20-19. I2CMDR Register Field Descriptions....2379
Table 20-20. I2CISRC Register Field Descriptions....2383
Table 20-21. I2CEMDR Register Field Descriptions....2384
Table 20-22. I2CPSC Register Field Descriptions....2385
Table 20-23. I2CFFTX Register Field Descriptions....2386
Table 20-24. I2CFFRX Register Field Descriptions....2388
Table 20-25. I2C Registers to Driverlib Functions....2389
Table 21-1. McBSP Interface Pins/Signals....2393
Table 21-2. Register Bits That Determine the Number of Phases, Words, and Bits....2400
Table 21-3. Interrupts and DMA Events Generated by a McBSP....2404
Table 21-4. Effects of DLB and CLKSTP on Clock Modes....2406
Table 21-5. Choosing an Input Clock for the Sample Rate Generator with the SCLKME and CLKSM Bits....2406
Table 21-6. Polarity Options for the Input to the Sample Rate Generator....2407
Table 21-7. Input Clock Selection for Sample Rate Generator....2411
Table 21-8. Block - Channel Assignment....2420
Table 21-9. 2-Partition Mode....2420
Table 21-10. 8-Partition Mode....2420
Table 21-11. Receive Channel Assignment and Control With Eight Receive Partitions....2423
Table 21-12. Transmit Channel Assignment and Control When Eight Transmit Partitions Are Used....2423
Table 21-13. Selecting a Transmit Multichannel Selection Mode With the XMCM Bits....2425
Table 21-14. Bits Used to Enable and Configure the Clock Stop Mode....2429
Table 21-15. Effects of CLKSTP, CLKXP, and CLKRP on the Clock Scheme....2429



Copyright © 2024 Texas Instruments Incorporated

Table 21-16. Bit Values Required to Configure the McBSP as an SPI Master ..... 2433
Table 21-17. Bit Values Required to Configure the McBSP as an SPI Slave ..... 2434
Table 21-18. Register Bits Used to Reset or Enable the McBSP Receiver Field Descriptions..... 2436
Table 21-19. Reset State of Each McBSP Pin..... 2436
Table 21-20. Register Bit Used to Enable/Disable the Digital Loopback Mode..... 2437
Table 21-21. Receive Signals Connected to Transmit Signals in Digital Loopback Mode..... 2437
Table 21-22. Register Bits Used to Enable/Disable the Clock Stop Mode..... 2437
Table 21-23. Effects of CLKSTP, CLKXP, and CLKRP on the Clock Scheme..... 2438
Table 21-24. Register Bit Used to Enable/Disable the Receive Multichannel Selection Mode..... 2438
Table 21-25. Register Bit Used to Choose One or Two Phases for the Receive Frame..... 2438
Table 21-26. Register Bits Used to Set the Receive Word Lengths..... 2439
Table 21-27. Register Bits Used to Set the Receive Frame Length..... 2440
Table 21-28. How to Calculate the Length of the Receive Frame..... 2440
Table 21-29. Register Bit Used to Enable/Disable the Receive Frame-Synchronization Ignore Function..... 2441
Table 21-30. Register Bits Used to Set the Receive Companding Mode..... 2442
Table 21-31. Register Bits Used to Set the Receive Data Delay..... 2444
Table 21-32. Register Bits Used to Set the Receive Sign-Extension and Justification Mode..... 2446
Table 21-33. Example: Use of RJUST Field With 12-Bit Data Value ABCh..... 2446
Table 21-34. Example: Use of RJUST Field With 20-Bit Data Value ABCDEh..... 2446
Table 21-35. Register Bits Used to Set the Receive Interrupt Mode..... 2447
Table 21-36. Register Bits Used to Set the Receive Frame Synchronization Mode..... 2448
Table 21-37. Select Sources to Provide the Receive Frame-Synchronization Signal and the Effect on the FSR Pin..... 2449
Table 21-38. Register Bit Used to Set Receive Frame-Synchronization Polarity..... 2449
Table 21-39. Register Bits Used to Set the SRG Frame-Synchronization Period and Pulse Width..... 2451
Table 21-40. Register Bits Used to Set the Receive Clock Mode..... 2452
Table 21-41. Receive Clock Signal Source Selection..... 2453
Table 21-42. Register Bit Used to Set Receive Clock Polarity..... 2453
Table 21-43. Register Bits Used to Set the Sample Rate Generator (SRG) Clock Divide-Down Value..... 2455
Table 21-44. Register Bit Used to Set the SRG Clock Synchronization Mode..... 2455
Table 21-45. Register Bits Used to Set the SRG Clock Mode (Choose an Input Clock)..... 2456
Table 21-46. Register Bits Used to Set the SRG Input Clock Polarity..... 2456
Table 21-47. Register Bits Used to Place Transmitter in Reset Field Descriptions..... 2458
Table 21-48. Register Bit Used to Enable/Disable the Digital Loopback Mode..... 2459
Table 21-49. Receive Signals Connected to Transmit Signals in Digital Loopback Mode..... 2459
Table 21-50. Register Bits Used to Enable/Disable the Clock Stop Mode..... 2459
Table 21-51. Effects of CLKSTP, CLKXP, and CLKRP on the Clock Scheme..... 2460
Table 21-52. Register Bits Used to Enable/Disable Transmit Multichannel Selection..... 2460
Table 21-53. Use of the Transmit Channel Enable Registers..... 2461
Table 21-54. Register Bit Used to Choose 1 or 2 Phases for the Transmit Frame..... 2463
Table 21-55. Register Bits Used to Set the Transmit Word Lengths..... 2463
Table 21-56. Register Bits Used to Set the Transmit Frame Length..... 2464
Table 21-57. How to Calculate Frame Length..... 2464
Table 21-58. Register Bit Used to Enable/Disable the Transmit Frame-Synchronization Ignore Function..... 2465
Table 21-59. Register Bits Used to Set the Transmit Companding Mode..... 2466
Table 21-60. Register Bits Used to Set the Transmit Data Delay..... 2467
Table 21-61. Register Bit Used to Set the Transmit DXENA (DX Delay Enabler) Mode..... 2469
Table 21-62. Register Bits Used to Set the Transmit Interrupt Mode..... 2469
Table 21-63. Register Bits Used to Set the Transmit Frame-Synchronization Mode..... 2470
Table 21-64. How FSXM and FSGM Select the Source of Transmit Frame-Synchronization Pulses..... 2470
Table 21-65. Register Bit Used to Set Transmit Frame-Synchronization Polarity..... 2471
Table 21-66. Register Bits Used to Set SRG Frame-Synchronization Period and Pulse Width..... 2472
Table 21-67. Register Bit Used to Set the Transmit Clock Mode..... 2473
Table 21-68. How the CLKXM Bit Selects the Transmit Clock and the Corresponding Status of the MCLKX pin..... 2473
Table 21-69. Register Bit Used to Set Transmit Clock Polarity..... 2473
Table 21-70. McBSP Emulation Modes Selectable with FREE and SOFT Bits of SPCR2..... 2475
Table 21-71. Reset State of Each McBSP Pin..... 2475
Table 21-72. Receive Interrupt Sources and Signals..... 2481
Table 21-73. Transmit Interrupt Sources and Signals..... 2482
Table 21-74. Error Flags..... 2482
Table 21-75. McBSP Mode Selection..... 2483
Table 21-76. McBSP Base Address Table..... 2485



Copyright © 2024 Texas Instruments Incorporated

Table 21-77. MCBSP_REGS Registers....2486
Table 21-78. McBSP_REGS Access Type Codes....2486
Table 21-79. DRR2 Register Field Descriptions....2488
Table 21-80. DRR1 Register Field Descriptions....2489
Table 21-81. DXR2 Register Field Descriptions....2490
Table 21-82. DXR1 Register Field Descriptions....2491
Table 21-83. SPCR2 Register Field Descriptions....2492
Table 21-84. SPCR1 Register Field Descriptions....2495
Table 21-85. RCR2 Register Field Descriptions....2498
Table 21-86. RCR1 Register Field Descriptions....2500
Table 21-87. XCR2 Register Field Descriptions....2501
Table 21-88. XCR1 Register Field Descriptions....2503
Table 21-89. SRGR2 Register Field Descriptions....2504
Table 21-90. SRGR1 Register Field Descriptions....2506
Table 21-91. MCR2 Register Field Descriptions....2507
Table 21-92. MCR1 Register Field Descriptions....2509
Table 21-93. RCERA Register Field Descriptions....2511
Table 21-94. RCERB Register Field Descriptions....2512
Table 21-95. XCERA Register Field Descriptions....2513
Table 21-96. XCERB Register Field Descriptions....2514
Table 21-97. PCR Register Field Descriptions....2515
Table 21-98. RCERC Register Field Descriptions....2518
Table 21-99. RCERD Register Field Descriptions....2519
Table 21-100. XCERC Register Field Descriptions....2520
Table 21-101. XCERD Register Field Descriptions....2521
Table 21-102. RCERE Register Field Descriptions....2522
Table 21-103. RCERF Register Field Descriptions....2523
Table 21-104. XCERE Register Field Descriptions....2524
Table 21-105. XCERF Register Field Descriptions....2525
Table 21-106. RCERG Register Field Descriptions....2526
Table 21-107. RCERH Register Field Descriptions....2527
Table 21-108. XCERG Register Field Descriptions....2528
Table 21-109. XCERH Register Field Descriptions....2529
Table 21-110. MFFINT Register Field Descriptions....2530
Table 21-111. MCBSP Registers to Driverlib Functions....2530
Table 22-1. CAN Register Access from Software....2539
Table 22-2. CAN Register Access from Code Composer Studio™ IDE....2539
Table 22-3. PIE Module Nomenclature for Interrupts....2547
Table 22-4. Programmable Ranges Required by CAN Protocol....2559
Table 22-5. Message Object Field Descriptions....2569
Table 22-6. Message RAM Addressing in Debug Mode....2572
Table 22-7. CAN Base Address Table....2574
Table 22-8. CAN_REGS Registers....2575
Table 22-9. CAN_REGS Access Type Codes....2576
Table 22-10. CAN_CTL Register Field Descriptions....2577
Table 22-11. CAN_ES Register Field Descriptions....2580
Table 22-12. CAN_ERRC Register Field Descriptions....2582
Table 22-13. CAN_BTR Register Field Descriptions....2583
Table 22-14. CAN_INT Register Field Descriptions....2585
Table 22-15. CAN_TEST Register Field Descriptions....2586
Table 22-16. CAN_PERR Register Field Descriptions....2588
Table 22-17. CAN_RAM_INIT Register Field Descriptions....2589
Table 22-18. CAN_GLB_INT_EN Register Field Descriptions....2590
Table 22-19. CAN_GLB_INT_FLG Register Field Descriptions....2591
Table 22-20. CAN_GLB_INT_CLR Register Field Descriptions....2592
Table 22-21. CAN_ABOTR Register Field Descriptions....2593
Table 22-22. CAN_TXRQ_X Register Field Descriptions....2594
Table 22-23. CAN_TXRQ_21 Register Field Descriptions....2595
Table 22-24. CAN_NDAT_X Register Field Descriptions....2596
Table 22-25. CAN_NDAT_21 Register Field Descriptions....2597
Table 22-26. CAN_IPEN_X Register Field Descriptions....2598



Copyright © 2024 Texas Instruments Incorporated

Table 22-27. CAN_IPEN_21 Register Field Descriptions....2599
Table 22-28. CAN_MVAL_X Register Field Descriptions....2600
Table 22-29. CAN_MVAL_21 Register Field Descriptions....2601
Table 22-30. CAN_IP_MUX21 Register Field Descriptions....2602
Table 22-31. CAN_IF1CMD Register Field Descriptions....2603
Table 22-32. CAN_IF1MSK Register Field Descriptions....2606
Table 22-33. CAN_IF1ARB Register Field Descriptions....2607
Table 22-34. CAN_IF1MCTL Register Field Descriptions....2609
Table 22-35. CAN_IF1DATA Register Field Descriptions....2611
Table 22-36. CAN_IF1DATB Register Field Descriptions....2612
Table 22-37. CAN_IF2CMD Register Field Descriptions....2613
Table 22-38. CAN_IF2MSK Register Field Descriptions....2616
Table 22-39. CAN_IF2ARB Register Field Descriptions....2617
Table 22-40. CAN_IF2MCTL Register Field Descriptions....2619
Table 22-41. CAN_IF2DATA Register Field Descriptions....2621
Table 22-42. CAN_IF2DATB Register Field Descriptions....2622
Table 22-43. CAN_IF3OBS Register Field Descriptions....2623
Table 22-44. CAN_IF3MSK Register Field Descriptions....2625
Table 22-45. CAN_IF3ARB Register Field Descriptions....2626
Table 22-46. CAN_IF3MCTL Register Field Descriptions....2627
Table 22-47. CAN_IF3DATA Register Field Descriptions....2629
Table 22-48. CAN_IF3DATB Register Field Descriptions....2630
Table 22-49. CAN_IF3UPD Register Field Descriptions....2631
Table 22-50. CAN Registers to Driverlib Functions....2631
Table 23-1. USB Memory Access from Software....2649
Table 23-2. USB Memory Access from CCS IDE....2650
Table 23-3. USB Base Address Table (C28)....2652
Table 23-4. Universal Serial Bus (USB) Controller Register Map....2652
Table 23-5. USB Device Functional Address Register (USBFADDR) Field Descriptions....2660
Table 23-6. USB Power Management Register (USBPOWER) in Host Mode Field Descriptions....2661
Table 23-7. USB Power Management Register (USBPOWER) in Device Mode Field Descriptions....2662
Table 23-8. USB Transmit Interrupt Status Register (USBTXIS) Field Descriptions....2663
Table 23-9. USB Transmit Interrupt Status Register (USBRXIS) Field Descriptions....2665
Table 23-10. USB Transmit Interrupt Status Register (USBTXIE) Field Descriptions....2667
Table 23-11. USB Transmit Interrupt Status Register (USBRXIE) Field Descriptions....2669
Table 23-12. USB General Interrupt Status Register (USBIS) in Host Mode Field Descriptions....2671
Table 23-13. USB General Interrupt Status Register (USBIS) in Device Mode Field Descriptions....2672
Table 23-14. USB Interrupt Enable Register (USBIE) in Host Mode Field Descriptions....2673
Table 23-15. USB Interrupt Enable Register (USBIE) in Device Mode Field Descriptions....2674
Table 23-16. USB Frame Value Register (USBFRAME) Field Descriptions....2675
Table 23-17. USB Endpoint Index Register (USBEPIDX) Field Descriptions....2675
Table 23-18. USB Test Mode Register (USBTEST) in Host Mode Field Descriptions....2676
Table 23-19. USB Test Mode Register (USBTEST) in Device Mode Field Descriptions....2677
Table 23-20. USB FIFO Endpoint *n* Register (USBFIFO[*n*]) Field Descriptions....2678
Table 23-21. USB Device Control Register (USBDEVCTL) Field Descriptions....2679
Table 23-22. USB Transmit Dynamic FIFO Sizing Register (USBTXFIFOSZ) Field Descriptions....2681
Table 23-23. USB Receive Dynamic FIFO Sizing Register (USBRXFIFOSZ) Field Descriptions....2682
Table 23-24. USB Transmit FIFO Start Address Register (USBTXFIFOADDR) Field Descriptions....2683
Table 23-25. USB Receive FIFO Start Address Register (USBRXFIFOADDR) Field Descriptions....2684
Table 23-26. USB Connect Timing Register (USBCONTIM) Field Descriptions....2685
Table 23-27. USB Full-Speed Last Transaction to End of Frame Timing Register (USBFSEOF) Field Descriptions....2686
Table 23-28. USB Low-Speed Last Transaction to End of Frame Timing Register (USBLSEOF) Field Descriptions....2686
Table 23-29. USB Transmit Functional Address Endpoint *n* Registers (USBTXFUNCADDR[*n*]) Field Descriptions....2687
Table 23-30. USB Transmit Hub Address Endpoint *n* Registers(USBTXHUBADDR[*n*]) Field Descriptions....2688
Table 23-31. USB Transmit Hub Port Endpoint *n* Registers(USBTXHUBPORT[*n*]) Field Descriptions....2689
Table 23-32. USB Receive Functional Address Endpoint *n* Registers(USBFIFO[*n*]) Field Descriptions....2690
Table 23-33. USB Receive Hub Address Endpoint *n* Registers (USBRXHUBADDR[*n*]) Field Descriptions....2691
Table 23-34. USB Receive Hub Port Endpoint *n* Register (USBRXHUBPORT[*n*]) Field Descriptions....2692
Table 23-35. USB Control and Status Endpoint 0 Low Register (USBCSRL0) in Host Mode Field Descriptions....2693
Table 23-36. USB Control and Status Endpoint 0 Low Register (USBCSRL0) in Device Mode Field Descriptions....2694
Table 23-37. USB Control and Status Endpoint 0 High Register (USBCSRH0) in Host Mode Field Descriptions....2696



Copyright © 2024 Texas Instruments Incorporated

Table 23-38. USB Control and Status Endpoint 0 High Register (USBCSRH0) in Device Mode Field Descriptions..... 2697
Table 23-39. USB Receive Byte Count Endpoint 0 Register (USBCOUNT0) Field Descriptions..... 2698
Table 23-40. USB Type Endpoint 0 Register (USBTYPE0) Field Descriptions..... 2698
Table 23-41. USB NAK Limit Register (USBNAKLMT) Field Descriptions..... 2699
Table 23-42. USB Maximum Transmit Data Endpoint *n* Registers (USBTXMAXP[*n*]) Field Descriptions..... 2700
Table 23-43. USB Transmit Control and Status Endpoint n Low Register (USBTXCSRL[*n*]) in Host Mode Field Descriptions..... 2701
Table 23-44. USB Transmit Control and Status Endpoint n Low Register (USBTXCSRL[*n*]) in Device Mode Field Descriptions..... 2702
Table 23-45. USB Transmit Control and Status Endpoint n High Register (USBTXCSRH[*n*]) in Host Mode Field Descriptions..... 2704
Table 23-46. USB Transmit Control and Status Endpoint n High Register (USBTXCSRH[*n*]) in Device Mode Field Descriptions..... 2705
Table 23-47. USB Maximum Receive Data Endpoint *n* Registers (USBRXMAXP[*n*]) Field Descriptions..... 2706
Table 23-48. USB Receive Control and Status Endpoint n Low Register (USBRXCSRL[*n*]) in Host Mode Field Descriptions..... 2707
Table 23-49. USB Receive Control and Status Endpoint n Low Register (USBRX CSRL[*n*]) in Device Mode Field Descriptions..... 2708
Table 23-50. USB Receive Control and Status Endpoint n High Register (USBRXCSRH[*n*]) in Host Mode Field Descriptions..... 2710
Table 23-51. USB Receive Control and Status Endpoint n High Register (USBRXCSRH[*n*]) in Device Mode Field Descriptions..... 2711
Table 23-52. USB Receive Byte Count Endpoint n Register (USBRXCOUNT[*n*]) Field Descriptions..... 2712
Table 23-53. USB Host Transmit Configure Type Endpoint n Registers (USBTXTYPE[*n*]) Field Descriptions..... 2713
Table 23-54. USB Host Transmit Interval Endpoint *n* Registers (USBTXINTERVAL[*n*]) Frame Numbers..... 2714
Table 23-55. USB Host Transmit Interval Endpoint *n* Registers (USBTXINTERVAL[*n*]) Field Descriptions..... 2714
Table 23-56. USB Host Configure Receive Type Endpoint n Register (USBRXTYPE[*n*]) Field Descriptions..... 2715
Table 23-57. USB Host Receive Polling Interval Endpoint *n* Registers (USBRXINTERVAL[*n*]) Frame Numbers..... 2716
Table 23-58. USB Host Receive Polling Interval Endpoint *n* Registers (USBRXINTERVAL[*n*]) Field Descriptions..... 2716
Table 23-59. USB Request Packet Count in Block Transfer Endpoint *n* Registers (USBRQPKTCOUNT[*n*]) Field Descriptions..... 2717
Table 23-60. USB Receive Double Packet Buffer Disable Register (USBRXDPKTBUFDIS) Field Descriptions..... 2718
Table 23-61. USB Transmit Double Packet Buffer Disable Register (USBTXDPKTBUFDIS) Field Descriptions..... 2719
Table 23-62. USB External Power Control Register (USBEPC) Field Descriptions..... 2720
Table 23-63. USB External Power Control Raw Interrupt Status Register (USBEPCRIS) Field Descriptions..... 2722
Table 23-64. USB External Power Control Interrupt Mask Register (USBEPCIM) Field Descriptions..... 2723
Table 23-65. USB External Power Control Interrupt Status and Clear Register (USBEPCISC) Field Descriptions..... 2724
Table 23-66. USB Device RESUME Raw Interrupt Status Register (USBDRRIS) Field Descriptions..... 2725
Table 23-67. USB Device RESUME Raw Interrupt Status Register (USBDRRIS) Field Descriptions..... 2726
Table 23-68. USB Device RESUME Interrupt Status and Clear Register (USBDRISC) Field Descriptions..... 2727
Table 23-69. USB General-Purpose Control and Status Register (USBGPCS) Field Descriptions..... 2728
Table 23-70. USB DMA Select Register (USBDMASEL) Field Descriptions..... 2729
Table 23-71. USB Registers to Driverlib Functions..... 2731
Table 24-1. uPP Signal Description..... 2753
Table 24-2. CPU/CLA/uPP-DMA Address Map..... 2761
Table 24-3. CPU/CLA/uPP-DMA Address Map..... 2761
Table 24-4. uPP Parameters Useful for System Tuning..... 2763
Table 24-5. UPP Base Address Table..... 2763
Table 24-6. UPP_REGS Registers..... 2764
Table 24-7. UPP_REGS Access Type Codes..... 2764
Table 24-8. PID Register Field Descriptions..... 2766
Table 24-9. PERCTL Register Field Descriptions..... 2767
Table 24-10. CHCTL Register Field Descriptions..... 2769
Table 24-11. IFCFG Register Field Descriptions..... 2770
Table 24-12. IFIVAL Register Field Descriptions..... 2772
Table 24-13. THCFG Register Field Descriptions..... 2773
Table 24-14. RAWINTST Register Field Descriptions..... 2775
Table 24-15. ENINTST Register Field Descriptions..... 2777
Table 24-16. INTENSET Register Field Descriptions..... 2779
Table 24-17. INTENCLR Register Field Descriptions..... 2781
Table 24-18. CHIDESC0 Register Field Descriptions..... 2783



Copyright © 2024 Texas Instruments Incorporated

Table 24-19. CHIDESC1 Register Field Descriptions....2784
Table 24-20. CHIDESC2 Register Field Descriptions....2785
Table 24-21. CHIST0 Register Field Descriptions....2786
Table 24-22. CHIST1 Register Field Descriptions....2787
Table 24-23. CHIST2 Register Field Descriptions....2788
Table 24-24. CHQDESC0 Register Field Descriptions....2789
Table 24-25. CHQDESC1 Register Field Descriptions....2790
Table 24-26. CHQDESC2 Register Field Descriptions....2791
Table 24-27. CHQST0 Register Field Descriptions....2792
Table 24-28. CHQST1 Register Field Descriptions....2793
Table 24-29. CHQST2 Register Field Descriptions....2794
Table 24-30. GINTEN Register Field Descriptions....2795
Table 24-31. GINTFLG Register Field Descriptions....2796
Table 24-32. GINTCLR Register Field Descriptions....2797
Table 24-33. DLYCTL Register Field Descriptions....2798
Table 24-34. UPP Registers to Driverlib Functions....2798
Table 25-1. Configuration for EMIF1 and EMIF2 Modules....2802
Table 25-2. EMIF Pins Used to Access Both SDRAM and Asynchronous Memories....2807
Table 25-3. EMIF Pins Specific to SDRAM....2807
Table 25-4. EMIF Pins Specific to Asynchronous Memory....2808
Table 25-5. EMIF SDRAM Commands....2808
Table 25-6. Truth Table for SDRAM Commands....2809
Table 25-7. 16-bit EMIF Address Pin Connections....2811
Table 25-8. Description of the SDRAM Configuration Register (SDRAM_CR)....2812
Table 25-9. Description of the SDRAM Refresh Control Register (SDRAM_RCR)....2812
Table 25-10. Description of the SDRAM Timing Register (SDRAM_TR)....2812
Table 25-11. Description of the SDRAM Self Refresh Exit Timing Register (SDR_EXT_TMNG)....2813
Table 25-12. SDRAM LOAD MODE REGISTER Command....2813
Table 25-13. Refresh Urgency Levels....2815
Table 25-14. Mapping from Logical Address to EMIF Pins for 32-bit SDRAM....2820
Table 25-15. Mapping from Logical Address to EMIF Pins for 16-bit SDRAM....2820
Table 25-16. Normal Mode vs. Select Strobe Mode....2821
Table 25-17. Description of the Asynchronous *m* Configuration Register (ASYNC_CSn_CR)....2823
Table 25-18. Description of the Asynchronous Wait Cycle Configuration Register (ASYNC_WCCR)....2824
Table 25-19. Description of EMIF Interrupt Mask Set Register (INT_MSK_SET)....2824
Table 25-20. Description of EMIF Interrupt Mast Clear Register (INT_MSK_CLR)....2825
Table 25-21. Asynchronous Read Operation in Normal Mode....2825
Table 25-22. Asynchronous Write Operation in Normal Mode....2827
Table 25-23. Asynchronous Read Operation in Select Strobe Mode....2829
Table 25-24. Asynchronous Write Operation in Select Strobe Mode....2831
Table 25-25. Interrupt Monitor and Control Bit Fields....2834
Table 25-26. SR Field Value For EMIF to K4S641632H-TC(L)70 Interface....2837
Table 25-27. SDRAM_TR Field Calculations for EMIF to K4S641632H-TC(L)70 Interface....2838
Table 25-28. RR Calculation for EMIF to K4S641632H-TC(L)70 Interface....2839
Table 25-29. RR Calculation for EMIF to K4S641632H-TC(L)70 Interface....2839
Table 25-30. SDRAM_CR Field Values For EMIF to K4S641632H-TC(L)70 Interface....2840
Table 25-31. AC Characteristics for a Read Access....2841
Table 25-32. AC Characteristics for a Write Access....2841
Table 25-33. EMIF Base Address Table....2844
Table 25-34. EMIF_REGS Registers....2845
Table 25-35. EMIF_REGS Access Type Codes....2845
Table 25-36. RCSR Register Field Descriptions....2846
Table 25-37. ASYNC_WCCR Register Field Descriptions....2847
Table 25-38. SDRAM_CR Register Field Descriptions....2848
Table 25-39. SDRAM_RCR Register Field Descriptions....2850
Table 25-40. ASYNC_CS2_CR Register Field Descriptions....2851
Table 25-41. ASYNC_CS3_CR Register Field Descriptions....2853
Table 25-42. ASYNC_CS4_CR Register Field Descriptions....2855
Table 25-43. SDRAM_TR Register Field Descriptions....2857
Table 25-44. TOTAL_SDRAM_AR Register Field Descriptions....2858
Table 25-45. TOTAL_SDRAM_ACTR Register Field Descriptions....2859



Copyright © 2024 Texas Instruments Incorporated

Table 25-46. SDR_EXT_TMNG Register Field Descriptions....2860
Table 25-47. INT_RAW Register Field Descriptions....2861
Table 25-48. INT_MSK Register Field Descriptions....2862
Table 25-49. INT_MSK_SET Register Field Descriptions....2863
Table 25-50. INT_MSK_CLR Register Field Descriptions....2864
Table 25-51. EMIF1_CONFIG_REGS Registers....2865
Table 25-52. EMIF1_CONFIG_REGS Access Type Codes....2865
Table 25-53. EMIF1LOCK Register Field Descriptions....2866
Table 25-54. EMIF1COMMIT Register Field Descriptions....2867
Table 25-55. EMIF1MSEL Register Field Descriptions....2868
Table 25-56. EMIF1ACCPROT0 Register Field Descriptions....2869
Table 25-57. EMIF2_CONFIG_REGS Registers....2870
Table 25-58. EMIF2_CONFIG_REGS Access Type Codes....2870
Table 25-59. EMIF2LOCK Register Field Descriptions....2871
Table 25-60. EMIF2COMMIT Register Field Descriptions....2872
Table 25-61. EMIF2ACCPROT0 Register Field Descriptions....2873
Table 25-62. EMIF Registers to Driverlib Functions....2873
Table 26-1. Global Signals and Mux Selection....2881
Table 26-2. Local Signals and Mux Selection....2883
Table 26-3. CLB Output Signal Multiplexer Table....2886
Table 26-4. Output Table....2888
Table 26-5. Input Table....2889
Table 26-6. Ports Tied Off to Prevent Combinatorial Loops....2889
Table 26-7. Counter Block Operating Modes....2892
Table 26-8. HLC Event List....2897
Table 26-9. HLC Instruction Address Ranges....2898
Table 26-10. HLC Instruction Format....2898
Table 26-11. HLC Instruction Description....2898
Table 26-12. HLC Register Encoding....2899
Table 26-13. Non-Memory Mapped Register Addresses....2901
Table 26-14. CLB Base Address Table (C28)....2905
Table 26-15. CLB_LOGIC_CONFIG_REGS Registers....2906
Table 26-16. CLB_LOGIC_CONFIG_REGS Access Type Codes....2906
Table 26-17. CLB_COUNT_RESET Register Field Descriptions....2908
Table 26-18. CLB_COUNT_MODE_1 Register Field Descriptions....2909
Table 26-19. CLB_COUNT_MODE_0 Register Field Descriptions....2910
Table 26-20. CLB_COUNT_EVENT Register Field Descriptions....2911
Table 26-21. CLB_FSM_EXTRA_IN0 Register Field Descriptions....2912
Table 26-22. CLB_FSM_EXTERNAL_IN0 Register Field Descriptions....2913
Table 26-23. CLB_FSM_EXTERNAL_IN1 Register Field Descriptions....2914
Table 26-24. CLB_FSM_EXTRA_IN1 Register Field Descriptions....2915
Table 26-25. CLB_LUT4_IN0 Register Field Descriptions....2916
Table 26-26. CLB_LUT4_IN1 Register Field Descriptions....2917
Table 26-27. CLB_LUT4_IN2 Register Field Descriptions....2918
Table 26-28. CLB_LUT4_IN3 Register Field Descriptions....2919
Table 26-29. CLB_FSM_LUT_FN1_0 Register Field Descriptions....2920
Table 26-30. CLB_FSM_LUT_FN2 Register Field Descriptions....2921
Table 26-31. CLB_LUT4_FN1_0 Register Field Descriptions....2922
Table 26-32. CLB_LUT4_FN2 Register Field Descriptions....2923
Table 26-33. CLB_FSM_NEXT_STATE_0 Register Field Descriptions....2924
Table 26-34. CLB_FSM_NEXT_STATE_1 Register Field Descriptions....2925
Table 26-35. CLB_FSM_NEXT_STATE_2 Register Field Descriptions....2926
Table 26-36. CLB_MISC_CONTROL Register Field Descriptions....2927
Table 26-37. CLB_OUTPUT_LUT_0 Register Field Descriptions....2929
Table 26-38. CLB_OUTPUT_LUT_1 Register Field Descriptions....2930
Table 26-39. CLB_OUTPUT_LUT_2 Register Field Descriptions....2931
Table 26-40. CLB_OUTPUT_LUT_3 Register Field Descriptions....2932
Table 26-41. CLB_OUTPUT_LUT_4 Register Field Descriptions....2933
Table 26-42. CLB_OUTPUT_LUT_5 Register Field Descriptions....2934
Table 26-43. CLB_OUTPUT_LUT_6 Register Field Descriptions....2935
Table 26-44. CLB_OUTPUT_LUT_7 Register Field Descriptions....2936



Copyright © 2024 Texas Instruments Incorporated

Table 26-45. CLB_HLC_EVENT_SEL Register Field Descriptions....2937
Table 26-46. CLB_LOGIC_CONTROL_REGS Registers....2938
Table 26-47. CLB_LOGIC_CONTROL_REGS Access Type Codes....2938
Table 26-48. CLB_LOAD_EN Register Field Descriptions....2940
Table 26-49. CLB_LOAD_ADDR Register Field Descriptions....2941
Table 26-50. CLB_LOAD_DATA Register Field Descriptions....2942
Table 26-51. CLB_INPUT_FILTER Register Field Descriptions....2943
Table 26-52. CLB_IN_MUX_SEL_0 Register Field Descriptions....2945
Table 26-53. CLB_LCL_MUX_SEL_1 Register Field Descriptions....2947
Table 26-54. CLB_LCL_MUX_SEL_2 Register Field Descriptions....2948
Table 26-55. CLB_BUF_PTR Register Field Descriptions....2949
Table 26-56. CLB_GP_REG Register Field Descriptions....2950
Table 26-57. CLB_OUT_EN Register Field Descriptions....2951
Table 26-58. CLB_GLBL_MUX_SEL_1 Register Field Descriptions....2952
Table 26-59. CLB_GLBL_MUX_SEL_2 Register Field Descriptions....2953
Table 26-60. CLB_INTR_TAG_REG Register Field Descriptions....2954
Table 26-61. CLB_LOCK Register Field Descriptions....2955
Table 26-62. CLB_DBG_R0 Register Field Descriptions....2956
Table 26-63. CLB_DBG_R1 Register Field Descriptions....2957
Table 26-64. CLB_DBG_R2 Register Field Descriptions....2958
Table 26-65. CLB_DBG_R3 Register Field Descriptions....2959
Table 26-66. CLB_DBG_C0 Register Field Descriptions....2960
Table 26-67. CLB_DBG_C1 Register Field Descriptions....2961
Table 26-68. CLB_DBG_C2 Register Field Descriptions....2962
Table 26-69. CLB_DBG_OUT Register Field Descriptions....2963
Table 26-70. CLB_DATA_EXCHANGE_REGS Registers....2965
Table 26-71. CLB_DATA_EXCHANGE_REGS Access Type Codes....2965
Table 26-72. CLB_PUSH_y Register Field Descriptions....2966
Table 26-73. CLB_PULL_y Register Field Descriptions....2967
Table 26-74. CLB Registers to Driverlib Functions....2967



Copyright © 2024 Texas Instruments Incorporated

This page intentionally left blank.



Copyright © 2024 Texas Instruments Incorporated

## 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 sheet, rather a companion guide that should be used alongside the device-specific data sheet 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 sheet. This allows the data sheet to outline the high-level features of the device without unnecessary information about register descriptions or programming models.

**Note**

Texas Instruments is transitioning to use more inclusive terminology. Some language may be different than what you expect to see for certain technology areas.

## 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 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.



Copyright © 2024 Texas Instruments Incorporated

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™, Texas Instruments™, and controlSUITE™ are trademarks of Texas Instruments.
USB Specification Revision 2.0™ is a trademark of Compaq Computer Corp.
All trademarks are the property of their respective owners.



Copyright © 2024 Texas Instruments Incorporated

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


**1.1 Introduction** ..... 76
**1.2 C2000Ware Structure** ..... 76
**1.3 Documentation** ..... 76
**1.4 Devices** ..... 76
**1.5 Libraries** ..... 76
**1.6 Code Composer Studio™ Integrated Development Environment (IDE)** ..... 76
**1.7 SysConfig and PinMUX Tool** ..... 77





Copyright © 2024 Texas Instruments Incorporated

## 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

## 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.

## 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

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.



Copyright © 2024 Texas Instruments Incorporated

## 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



Copyright © 2024 Texas Instruments Incorporated

Chapter 2

# C28x Processor

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*


**2.1 Introduction** ..... 79
**2.2 C28X Related Collateral** ..... 79
**2.3 Features** ..... 79
**2.4 Floating-Point Unit** ..... 80
**2.5 Trigonometric Math Unit (TMU)** ..... 80
**2.6 Viterbi, Complex Math, and CRC Unit II (VCU-II)** ..... 81





Copyright © 2024 Texas Instruments Incorporated

## 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*
- How fast is your 32-bit MCU?

**Getting Started Materials**

- C2000 Multicore Development User Guide
- C2000 VCU, Viterbi, Complex Math, and CRC (Video)
- 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*
- *Migrating Software From 8-Bit (Byte) Addressable CPU's to C28x CPU Application Report*
- *TMS320C28x Extended Instruction Sets Application Report*
- *TMS320C28x FPU Primer Application Report*
- Texas Instruments F2837xD Peripheral Driver Library

## 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.



Copyright © 2024 Texas Instruments Incorporated

## 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 * 2pi | 2/3 |
| DIV2PIF32 RaH,RbH | a = b / 2pi | 2/3 |
| DIVF32 RaH,RbH,RcH | a = b/c | 5 |
| SQRTF32 RaH,RbH | a = sqrt(b) | 5 |
| SINPUF32 RaH,RbH | a = sin(b*2pi) | 4 |
| COSPUF32 RaH,RbH | a = cos(b*2pi) | 4 |
| ATANPUF32 RaH,RbH | a = atan(b)/2pi | 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*.



Copyright © 2024 Texas Instruments Incorporated

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

The VCU-II is the second-generation Viterbi, Complex Math, and CRC extension to the C28x CPU. The VCU-II extends the capabilities of the C28x CPU by adding registers and instructions to accelerate the performance of FFTs and communications-based algorithms. The C28x+VCU-II supports the following algorithm types:

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

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

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

(1) C28x CPU takes 15 cycles per butterfly.
(2) C28x CPU takes 22 cycles per stage.

- **Cyclic Redundancy Check:** Cyclic redundancy check (CRC) algorithms provide a straightforward method for verifying data integrity over large data blocks, communication packets, or code sections. The C28x+VCU can perform 8-bit, 16-bit, 24-bit, and 32-bit CRCs. For example, the VCU can compute the CRC for a block length of 10 bytes in 10 cycles. A CRC result register contains the current CRC, which is updated whenever a CRC instruction is executed.
- **Complex Math:** Complex math is used in many applications, a few are:
  - Fast Fourier Transform (FFT): The complex FFT is used in spread spectrum communications, as well as in many signal processing algorithms.
  - Complex filters: Complex filters improve data reliability, transmission distance, and power efficiency. The C28x+VCU can perform a complex I and Q multiply with coefficients (four multiplies) in a single cycle. In addition, the C28x+VCU can read/write the real and imaginary parts of 16-bit complex data to memory in a single cycle.

Table 2-3 shows a summary of a few complex math operations enabled by the VCU-II.

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

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

**Note**

Only the CRC-related VCU instructions are supported in future devices. FFT algorithms are available for the C28x+FPU.

For more information, see the *TMS320C28x Extended Instruction Sets Technical Reference Manual*.



Copyright © 2024 Texas Instruments Incorporated

*Chapter 3*

# *System Control and Interrupt*

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


**3.1 Introduction** ..... 83
**3.2 System Control Functional Description** ..... 83
**3.3 Resets** ..... 84
**3.4 Peripheral Interrupts** ..... 87
**3.5 Exceptions and Non-Maskable Interrupts** ..... 99
**3.6 Safety Features** ..... 101
**3.7 Clocking** ..... 104
**3.8 32-Bit CPU Timers 0/1/2** ..... 116
**3.9 Watchdog Timers** ..... 118
**3.10 Low-Power Modes** ..... 121
**3.11 Memory Controller Module** ..... 125
**3.12 Flash and OTP Memory** ..... 133
**3.13 Dual Code Security Module (DCSM)** ..... 147
**3.14 JTAG** ..... 160
**3.15 System Control Register Configuration Restrictions** ..... 160
**3.16 Software** ..... 161
**3.17 System Control Registers** ..... 166





Copyright © 2024 Texas Instruments Incorporated

## 3.1 Introduction

On this device, the CPU1 subsystem acts as a master, and by default (upon reset) owns all the configuration and control. Through software running on CPU1, peripherals and I/Os can be configured to be accessible by the CPU2 subsystem and the configuration chosen can be locked. All peripherals are available on CPU1, but not all peripherals are available on CPU2. To see what peripherals can be configured for CPU2, consult the CPUSELn registers in the DEV_CFG_REGS section.

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

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

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

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

This chapter explains the register space of the device system control module that is divided into three categories:

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

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

## 3.2 System Control Functional Description

The system control module provides the following capabilities:

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

### *3.2.1 Device Identification*

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

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

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



Copyright © 2024 Texas Instruments Incorporated

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

- UID_PSRAND0-5: 192 bits of pseudo-random data
- UID_UNIQUE: 32-bit unique data, the value in this register is unique across all devices with the same PARTIDH
- UID_CHECKSUM: 32-bit Fletcher checksum of UID_PSRAND0-5 and UID_UNIQUE
- CPU ID: 16-bit location in OTP memory. The value at this location provides the information about the CPU (CPU1 or CPU2). Refer to the device data sheet for more detail.

### *3.2.2 Device Configuration Registers*

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

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

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

This register is available for software to identify on which CPU it is executing.

## 3.3 Resets

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

### *3.3.1 Reset Sources*

Table 3-1 summarizes the various reset signals and their effect on the device.

**Table 3-1. Reset Signals**

| Reset Source | CPU1 Core Reset (C28x, TMU, FPU, VCU) | CPU1 Peripherals Reset | CPU2 Core Reset (C28x, TMU, FPU, VCU) | CPU2 Peripherals Reset | CPU2 Held In Reset | JTAG / Debug Logic Reset | IOs | $\overline{\text{XRS}}$ Output |
|---|---|---|---|---|---|---|---|---|
| POR | Yes | Yes | Yes | Yes | Yes | Yes | Hi-Z | Yes |
| $\overline{\text{XRS}}$ Pin | Yes | Yes | Yes | Yes | Yes | No | Hi-Z | - |
| CPU1.$\overline{\text{WDRS}}$ | Yes | Yes | Yes | Yes | Yes | No | Hi-Z | Yes |
| CPU1.$\overline{\text{NMIWDRS}}$ | Yes | Yes | Yes | Yes | Yes | No | Hi-Z | Yes |
| CPU1.$\overline{\text{SYSRS}}$ (Debugger Reset) | Yes | Yes | Yes | Yes | Yes | No | Hi-Z | No |
| CPU1.$\overline{\text{SCCRESET}}$ | Yes | Yes | Yes | Yes | Yes | No | Hi-Z | No |
| CPU2.$\overline{\text{SYSRS}}$ (Debugger Reset) | No | No | Yes | Yes | No | No | - | No |
| CPU2.$\overline{\text{WDRS}}$ | No | No | Yes | Yes | No | No | - | No |
| CPU2.$\overline{\text{NMIWDRS}}$ | No | No | Yes | Yes | No | No | - | No |
| CPU2.$\overline{\text{SCCRESET}}$ | No | No | Yes | Yes | No | No | - | No |
| $\overline{\text{HIBRESET}}$ | Yes | Yes | Yes | Yes | Yes | Yes | Isolated | No |
| CPU1.$\overline{\text{HWBISTRS}}$ | Yes | No | No | No | No | No | - | No |
| CPU2.$\overline{\text{HWBISTRS}}$ | No | No | Yes | No | No | No | - | No |
| $\overline{\text{TRST}}$ | No | No | No | No | No | Yes | - | No |



Copyright © 2024 Texas Instruments Incorporated

The resets can be divided into a few groups:

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

Whenever the CPU1 subsystem is reset, CPU2 and the peripherals are also reset, and CPU2 is held in reset. CPU1 brings CPU2 out of reset by writing to the CPU2RESCTL register. This is normally done by the boot ROM. For more details on the boot process, refer to Chapter 4.

After a reset, the reset cause register (RESC) is updated with the reset cause. The bits in this register maintain their state across multiple resets. The bits can only be cleared by a power-on reset (POR) or by writing ones to the register. Each CPU has their own RESC register, referred to as CPU1.RESC and CPU2.RESC.

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

**Note**

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

### *3.3.2 External Reset (XRS)*

The external reset (XRS) is the main chip-level reset for the device and resets both CPUs, all peripherals and I/O pin configurations, and most of the system control registers.XRS also holds CPU2 in reset. There is a dedicated open-drain pin for XRS. This pin can be used to drive reset pins for other ICs in the application, and can be driven by an external source. The XRS is driven internally during watchdog, NMI, and power-on resets. In hibernate mode, toggling XRS produces a HIBRESET.

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

### *3.3.3 Power-On Reset (POR)*

The power-on reset (POR) circuit creates a clean reset throughout the device during power-up, suppressing glitches on the GPIOs. The XRS pin is held low for the duration of the POR. In most applications, XRS is held low long enough to reset other system ICs, but some applications can require a longer pulse. In these cases, XRS can be driven low externally to provide the correct reset duration. A POR resets everything that XRS does, along with a few other registers – the reset cause register (RESC), the NMI shadow flag register (NMISHDFLG), the X1 clock counter register (X1CNT), and the hibernate configuration registers (HIBBOOTMODE, IORESTOREADDR, and LPMCR.M0M1MODE).

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

### *3.3.4 Debugger Reset (SYSRS)*

During development, it is sometimes necessary to reset the CPU and the peripherals without disconnecting the debugger or disrupting the system-level configuration. To facilitate this, each CPU has a subsystem reset, which can be triggered by a debugger using Code Composer Studio™ IDE. The CPU2 subsystem reset (CPU2.SYSRS) resets only CPU2, the peripherals, and the clock gating and LPM configuration. It does not hold CPU2 in reset. The CPU1 subsystem reset (CPU1.SYSRS) resets CPU1, the peripherals, many system control registers (including the clock gating and LPM configuration and the peripherals CPU ownership), and all I/O pin configurations. It also produces a CPU2.SYSRS and holds CPU2 in reset.



Copyright © 2024 Texas Instruments Incorporated

Neither $\overline{\text{SYSRS}}$ resets the ICEPick debug module, the device capability registers, the clock source and PLL configurations, the missing clock detection state, the PIE vector fetch error handler address, the NMI flags, the analog trims, or anything reset only by a POR (see Section 3.3.3).

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

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

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

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

Each CPU has a non-maskable interrupt (NMI) module that detects hardware errors in the system. Each NMI module has a watchdog timer that triggers a reset, if the CPU does not respond to an error within a user-specified amount of time. The CPU1 NMI watchdog reset (CPU1.$\overline{\text{NMIWDRS}}$) produces an $\overline{\text{XRS}}$. The CPU2 NMI watchdog reset (CPU2.$\overline{\text{NMIWDRS}}$) produces a CPU2.$\overline{\text{SYSRS}}$ and triggers an NMI on CPU1.

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

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

Each CPU has a dual-zone code security module (DCSM) that blocks read access to certain areas of the Flash memory. To facilitate CRC checks and copying of CLA code, TI provides ROM functions to securely access those memory areas. To prevent security breaches, interrupts must be disabled before calling these functions. If a vector fetch occurs in a safe copy or CRC function, the DCSM triggers a reset. The CPU1 security reset (CPU1.$\overline{\text{SCCRESET}}$) is similar to a CPU1.$\overline{\text{SYSRS}}$, and the CPU2 security reset (CPU2.$\overline{\text{SCCRESET}}$) is similar to a CPU2.$\overline{\text{SYSRS}}$. 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.

### 3.3.8 Hibernate Reset ($\overline{\text{HIBRESET}}$)

Hibernate is a chip-level low-power mode that gates power to large portions of the device. Waking up from hibernate involves a special reset ($\overline{\text{HIBRESET}}$). This reset is similar to a POR except that the I/O pins remain isolated and the $\overline{\text{XRS}}$ pin is not toggled. (An external $\overline{\text{XRS}}$ toggle during hibernate triggers a $\overline{\text{HIBRESET}}$). I/O isolation is disabled in software as part of a special boot ROM flow. For more information on hibernate, refer to Section 3.10.

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

### 3.3.9 Hardware BIST Reset ($\overline{\text{HWBISTRS}}$)

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

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

### 3.3.10 Test Reset ($\overline{\text{TRST}}$)

The ICEPick debug module and associated JTAG logic has a reset ($\overline{\text{TRST}}$) that is controlled by a dedicated pin. This reset is normally active unless the user connects a debugger to the device. For more information on the debug module, see the TI Processors Wiki page on ICEPick: http://processors.wiki.ti.com/index.php/ICEPICK.

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



Copyright © 2024 Texas Instruments Incorporated

## 3.4 Peripheral Interrupts

This section explains the peripheral interrupt handling on the device. Non-maskable interrupts are covered in Section 3.5. Software interrupts and emulation interrupts are not covered in this document. For information on those, see the *TMS320C28x CPU and Instruction Set Reference Guide*.

### *3.4.1 Interrupt Concepts*

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

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

### *3.4.2 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 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 16 peripheral interrupts into each CPU interrupt line and 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.

#### 3.4.2.1 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.

#### 3.4.2.2 PIE Stage

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

When the CPU receives an interrupt, 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 3.4.4 contains procedures for safely modifying the PIE configuration once interrupts have been enabled.



Copyright © 2024 Texas Instruments Incorporated

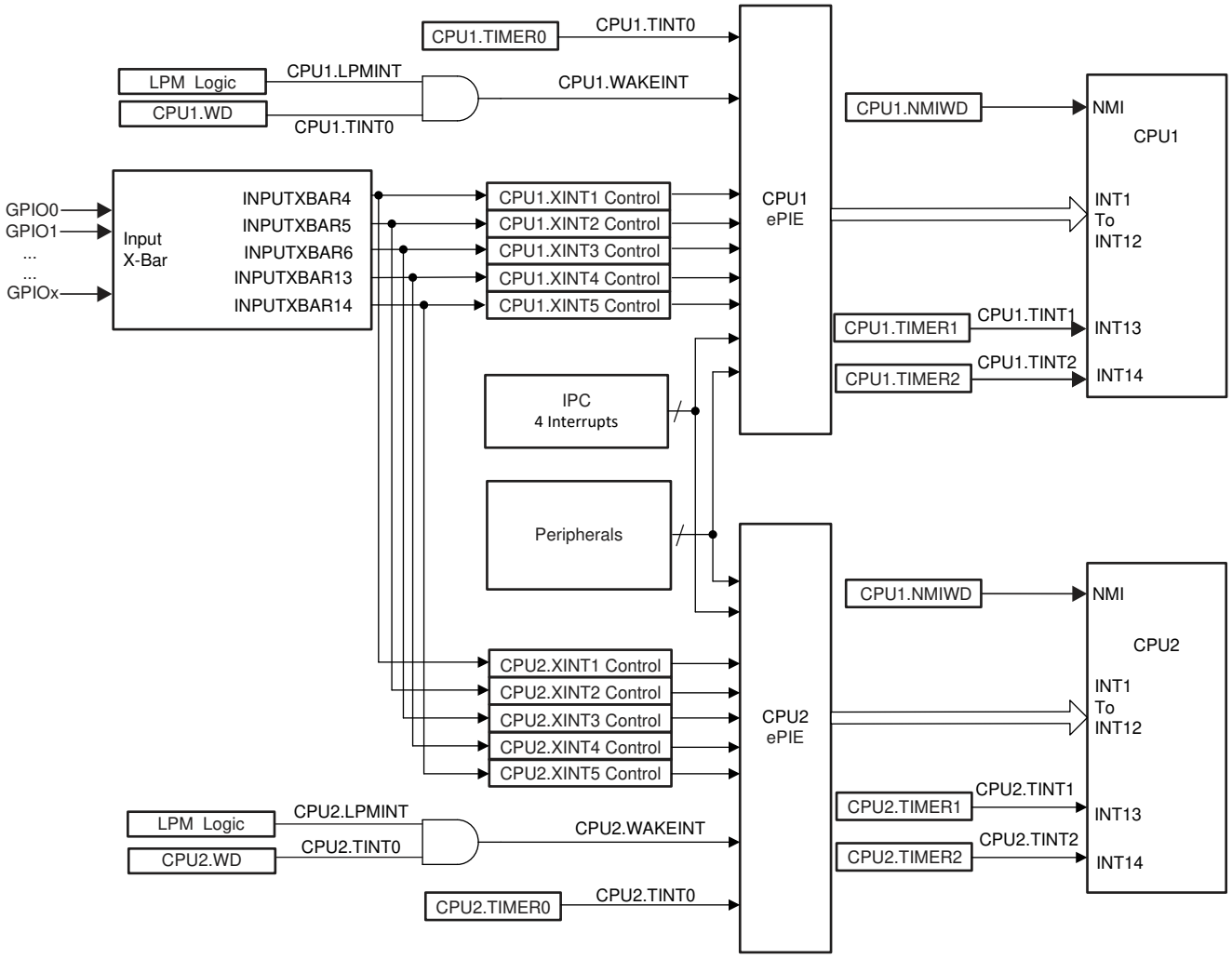



**Figure 3-1. Device Interrupt Architecture**

#### 3.4.2.3 CPU Stage

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

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

#### 3.4.2.4 Dual-CPU Interrupt Handling

Each CPU has a PIE. Both PIEs must be configured independently.

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



Copyright © 2024 Texas Instruments Incorporated

### 3.4.3 Interrupt Entry Sequence

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

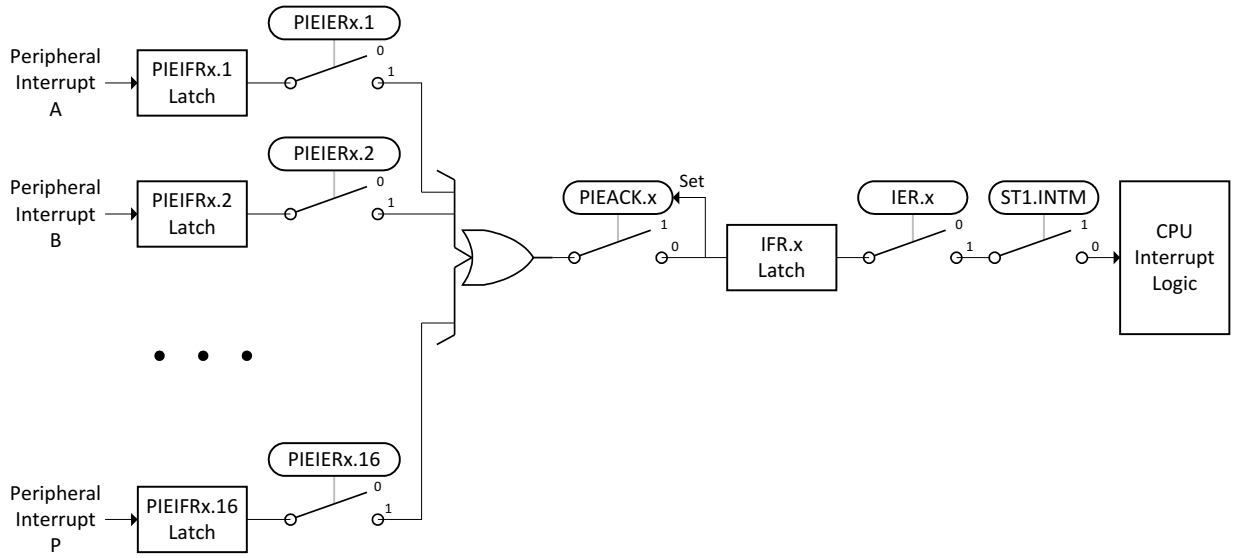



**Figure 3-2. Interrupt Propagation Path**

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

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 two SYSCLK cycles for GPIO synchronization plus extra time for input qualification (if used). Loops created using the RPT instruction cannot be interrupted.



Copyright © 2024 Texas Instruments Incorporated

### 3.4.4 Configuring and Using Interrupts

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

#### 3.4.4.1 Enabling Interrupts

To enable a peripheral interrupt, perform the following steps:

1. Disable interrupts globally (DINT or SETC INTM).
2. Enable the PIE by setting the ENPIE bit of the PIECTRL register.
3. Write the ISR vector for each interrupt to the appropriate location in the PIE vector table, which can be found in Section 3.4.6.
4. Set the appropriate PIEIERx bit for each interrupt. The PIE group and channel assignments can be found in Section 3.4.6.
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.

#### 3.4.4.2 Handling Interrupts

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

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

Requirements 1 and 3 are handled automatically by the TMS320C28x C compiler if the function is defined using the __interrupt keyword. For information on this keyword, see the Keywords section of the *TMS320C28x Optimizing C/C++ Compiler v6.2.4 User's Guide*. 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.



Copyright © 2024 Texas Instruments Incorporated

#### 3.4.4.3 Disabling Interrupts

To disable all interrupts, set the CPU's global interrupt mask using a 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, it may reach the CPU and trigger a spurious interrupt condition. To avoid this, use the following procedure:

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

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

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

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

#### 3.4.4.4 Nesting Interrupts

By default, interrupts do not nest. It is possible to nest and prioritize interrupts via software control of the IER and PIEIERx registers. Documentation and example code can be found in controlSUITE and on the TI Processors wiki:

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



Copyright © 2024 Texas Instruments Incorporated

### 3.4.5 PIE Channel Mapping

Table 3-2 shows the PIE group and channel assignments for each peripheral interrupt. Each row is a group, and each column is a channel within that group. When multiple interrupts are pending, the lowest-numbered channel is the lowest-numbered group is serviced first. Thus, the interrupts at the top of the table have the highest priority, and the interrupts at the bottom have the lowest priority.

**Note**

Cells marked "-" are Reserved

**Table 3-2. PIE Channel Mapping**

| | INTx.1 | INTx.2 | INTx.3 | INTx.4 | INTx.5 | INTx.6 | INTx.7 | INTx.8 | INTx.9 | INTx.10 | INTx.11 | INTx.12 | INTx.13 | INTx.14 | INTx.15 | INTx.16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| INT1.y | ADCA1 | ADCB1 | ADCC1 | XINT1 | XINT2 | ADCD1 | TIMER0 | WAKE | - | - | - | - | IPC1 | IPC2 | IPC3 | IPC4 |
| INT2.y | EPWM1_TZ | EPWM2_TZ | EPWM3_TZ | EPWM4_TZ | EPWM5_TZ | EPWM6_TZ | EPWM7_TZ | EPWM8_TZ | EPWM9_TZ | EPWM10_TZ | EPWM11_TZ | EPWM12_TZ | - | - | - | - |
| INT3.y | EPWM1 | EPWM2 | EPWM3 | EPWM4 | EPWM5 | EPWM6 | EPWM7 | EPWM8 | EPWM9 | EPWM10 | EPWM11 | EPWM12 | - | - | - | - |
| INT4.y | ECAP1 | ECAP2 | ECAP3 | ECAP4 | ECAP5 | ECAP6 | - | - | - | - | - | - | - | - | - | - |
| INT5.y | EQEP1 | EQEP2 | EQEP3 | - | CLB1 | CLB2 | CLB3 | CLB4 | SDFM1 | SDFM2 | - | - | - | - | - | - |
| INT6.y | SPIA_RX | SPIA_TX | SPIB_RX | SPIB_TX | MCBSPA_RX | MCBSPA_TX | MCBSPB_RX | MCBSPB_TX | SPIC_RX | SPIC_TX | - | - | - | - | - | - |
| INT7.y | DMA_CH1 | DMA_CH2 | DMA_CH3 | DMA_CH4 | DMA_CH5 | DMA_CH6 | - | - | - | - | - | - | - | - | - | - |
| INT8.y | I2CA | I2CA_FIFO | I2CB | I2CB_FIFO | SCIC_RX | SCIC_TX | SCID_RX | SCID_TX | - | - | - | - | - | - | UPPA | - |
| INT9.y | SCIA_RX | SCIA_TX | SCIB_RX | SCIB_TX | CANA_0 | CANA_1 | CANB_0 | CANB_1 | - | - | - | - | - | - | USBA | - |
| INT10.y | ADCA_EVT | ADCA2 | ADCA3 | ADCA4 | ADCB_EVT | ADCB2 | ADCB3 | ADCB4 | ADCC_EVT | ADCC2 | ADCC3 | ADCC4 | ADCD_EVT | ADCD2 | ADCD3 | ADCD4 |
| INT11.y | CLA1_1 | CLA1_2 | CLA1_3 | CLA1_4 | CLA1_5 | CLA1_6 | CLA1_7 | CLA1_8 | - | - | - | - | - | - | - | - |
| INT12.y | XINT3 | XINT4 | XINT5 | - | FMC | VCU | FPU OVERFLOW | FPU UNDERFLOW | EMIF_ERROR | RAM_CORRECTABLE_ERROR | FLASH_CORRECTABLE_ERROR | RAM_ACCESS_VIOLATION | SYS_PLL_SLIP | AUX_PLL_SLIP | CLA OVERFLOW | CLA UNDERFLOW |



Copyright © 2024 Texas Instruments Incorporated

#### 3.4.5.1 PIE Interrupt Priority

##### *3.4.5.1.1 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, in order for channel 1.3 to be serviced, channel 1.1's 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 channel 1.1's ISR completes, channel 1.2 which 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 remains pending. Using the steps from the Interrupt Entry Sequence (Section 3.4.3), 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 channel 1.2 is still serviced ahead of channel 1.3.

##### *3.4.5.1.2 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 their 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 3.4.3).

The following describes 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 3.4.3).

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 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 guaranteed if no interrupts are currently being serviced, that is, the Interrupt Entry Sequence (Section 3.4.3) is not executing.



Copyright © 2024 Texas Instruments Incorporated

### 3.4.6 Vector Tables

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

Table 3-4 shows the PIE vector table.

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

| Name | Vector ID | Address | Size (x16) | Description | Core priority | ePIE group Priority |
|---|---|---|---|---|---|---|
| Reset | 0 | 0x0000 0D00 | 2 | Reset is always fetched from location 0x003F_FFC0 in Boot ROM | 1 (Highest) | - |
| INT1 | 1 | 0x0000 0D02 | 2 | Not used. See PIE Group 1 | 5 | - |
| INT2 | 2 | 0x0000 0D04 | 2 | Not used. See PIE Group 2 | 6 | - |
| INT3 | 3 | 0x0000 0D06 | 2 | Not used. See PIE Group 3 | 7 | - |
| INT4 | 4 | 0x0000 0D08 | 2 | Not used. See PIE Group 4 | 8 | - |
| INT5 | 5 | 0x0000 0D0A | 2 | Not used. See PIE Group 5 | 9 | - |
| INT6 | 6 | 0x0000 0D0C | 2 | Not used. See PIE Group 6 | 10 | - |
| INT7 | 7 | 0x0000 0D0E | 2 | Not used. See PIE Group 7 | 11 | - |
| INT8 | 8 | 0x0000 0D10 | 2 | Not used. See PIE Group 8 | 12 | - |
| INT9 | 9 | 0x0000 0D12 | 2 | Not used. See PIE Group 9 | 13 | - |
| INT10 | 10 | 0x0000 0D14 | 2 | Not used. See PIE Group 10 | 14 | - |
| INT11 | 11 | 0x0000 0D16 | 2 | Not used. See PIE Group 11 | 15 | - |
| INT12 | 12 | 0x0000 0D18 | 2 | Not used. See PIE Group 12 | 16 | - |
| INT13 | 13 | 0x0000 0D1A | 2 | CPU TIMER1 Interrupt | 17 | - |
| INT14 | 14 | 0x0000 0D1C | 2 | CPU TIMER2 Interrupt (for TI/RTOS use) | 18 | - |
| DATALOG | 15 | 0x0000 0D1E | 2 | CPU Data Logging Interrupt | 19 (lowest) | - |
| RTOSINT | 16 | 0x0000 0D20 | 2 | CPU Real-Time OS Interrupt | 4 | - |
| 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 | - | - |



Copyright © 2024 Texas Instruments Incorporated

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

| Name | Vector ID | Address | Size (x16) | Description | Core Priority | ePIE Group Priority |
|---|---|---|---|---|---|---|
| **PIE Group 1 Vectors - Muxed into CPU INT1** | | | | | | |
| INT1.1 | 32 | 0x0000 0D40 | 2 | ADCA1 interrupt | 5 | 1 (Highest) |
| INT1.2 | 33 | 0x0000 0D42 | 2 | ADCB1 interrupt | 5 | 2 |
| INT1.3 | 34 | 0x0000 0D44 | 2 | ADCC1 interrupt | 5 | 3 |
| INT1.4 | 35 | 0x0000 0D46 | 2 | XINT1 interrupt | 5 | 4 |
| INT1.5 | 36 | 0x0000 0D48 | 2 | XINT2 interrupt | 5 | 5 |
| INT1.6 | 37 | 0x0000 0D4A | 2 | ADCD1 interrupt | 5 | 6 |
| INT1.7 | 38 | 0x0000 0D4C | 2 | TIMER0 interrupt | 5 | 7 |
| INT1.8 | 39 | 0x0000 0D4E | 2 | WAKE interrupt | 5 | 8 |
| INT1.9 | 128 | 0x0000 0E00 | 2 | Reserved | 5 | 9 |
| INT1.10 | 129 | 0x0000 0E02 | 2 | Reserved | 5 | 10 |
| INT1.11 | 130 | 0x0000 0E04 | 2 | Reserved | 5 | 11 |
| INT1.12 | 131 | 0x0000 0E06 | 2 | Reserved | 5 | 12 |
| INT1.13 | 132 | 0x0000 0E08 | 2 | IPC1 interrupt | 5 | 13 |
| INT1.14 | 133 | 0x0000 0E0A | 2 | IPC2 interrupt | 5 | 14 |
| INT1.15 | 134 | 0x0000 0E0C | 2 | IPC3 interrupt | 5 | 15 |
| INT1.16 | 135 | 0x0000 0E0E | 2 | IPC4 interrupt | 5 | 16 (Lowest) |
| **PIE Group 2 Vectors - Muxed into CPU INT2** | | | | | | |
| INT2.1 | 40 | 0x0000 0D50 | 2 | EPWM1_TZ interrupt | 6 | 1 (Highest) |
| INT2.2 | 41 | 0x0000 0D52 | 2 | EPWM2_TZ interrupt | 6 | 2 |
| INT2.3 | 42 | 0x0000 0D54 | 2 | EPWM3_TZ interrupt | 6 | 3 |
| INT2.4 | 43 | 0x0000 0D56 | 2 | EPWM4_TZ interrupt | 6 | 4 |
| INT2.5 | 44 | 0x0000 0D58 | 2 | EPWM5_TZ interrupt | 6 | 5 |
| INT2.6 | 45 | 0x0000 0D5A | 2 | EPWM6_TZ interrupt | 6 | 6 |
| INT2.7 | 46 | 0x0000 0D5C | 2 | EPWM7_TZ interrupt | 6 | 7 |
| INT2.8 | 47 | 0x0000 0D5E | 2 | EPWM8_TZ interrupt | 6 | 8 |
| INT2.9 | 136 | 0x0000 0E10 | 2 | EPWM9_TZ interrupt | 6 | 9 |
| INT2.10 | 137 | 0x0000 0E12 | 2 | EPWM10_TZ interrupt | 6 | 10 |
| INT2.11 | 138 | 0x0000 0E14 | 2 | EPWM11_TZ interrupt | 6 | 11 |
| INT2.12 | 139 | 0x0000 0E16 | 2 | EPWM12_TZ interrupt | 6 | 12 |
| INT2.13 | 140 | 0x0000 0E18 | 2 | Reserved | 6 | 13 |
| INT2.14 | 141 | 0x0000 0E1A | 2 | Reserved | 6 | 14 |
| INT2.15 | 142 | 0x0000 0E1C | 2 | Reserved | 6 | 15 |
| INT2.16 | 143 | 0x0000 0E1E | 2 | Reserved | 6 | 16 (Lowest) |
| **PIE Group 3 Vectors - Muxed into CPU INT3** | | | | | | |
| INT3.1 | 48 | 0x0000 0D60 | 2 | EPWM1 interrupt | 7 | 1 (Highest) |
| INT3.2 | 49 | 0x0000 0D62 | 2 | EPWM2 interrupt | 7 | 2 |
| INT3.3 | 50 | 0x0000 0D64 | 2 | EPWM3 interrupt | 7 | 3 |
| INT3.4 | 51 | 0x0000 0D66 | 2 | EPWM4 interrupt | 7 | 4 |
| INT3.5 | 52 | 0x0000 0D68 | 2 | EPWM5 interrupt | 7 | 5 |
| INT3.6 | 53 | 0x0000 0D6A | 2 | EPWM6 interrupt | 7 | 6 |
| INT3.7 | 54 | 0x0000 0D6C | 2 | EPWM7 interrupt | 7 | 7 |
| INT3.8 | 55 | 0x0000 0D6E | 2 | EPWM8 interrupt | 7 | 8 |
| INT3.9 | 144 | 0x0000 0E20 | 2 | EPWM9 interrupt | 7 | 9 |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-4. PIE Interrupt Vectors (continued)**

| Name | Vector ID | Address | Size (x16) | Description | Core Priority | ePIE Group Priority |
|---|---|---|---|---|---|---|
| INT3.10 | 145 | 0x0000 0E22 | 2 | EPWM10 interrupt | 7 | 10 |
| INT3.11 | 146 | 0x0000 0E24 | 2 | EPWM11 interrupt | 7 | 11 |
| INT3.12 | 147 | 0x0000 0E26 | 2 | EPWM12 interrupt | 7 | 12 |
| INT3.13 | 148 | 0x0000 0E28 | 2 | Reserved | 7 | 13 |
| INT3.14 | 149 | 0x0000 0E2A | 2 | Reserved | 7 | 14 |
| INT3.15 | 150 | 0x0000 0E2C | 2 | Reserved | 7 | 15 |
| INT3.16 | 151 | 0x0000 0E2E | 2 | Reserved | 7 | 16 (Lowest) |
| **PIE Group 4 Vectors - Muxed into CPU INT4** | | | | | | |
| INT4.1 | 56 | 0x0000 0D70 | 2 | ECAP1 interrupt | 8 | 1 (Highest) |
| INT4.2 | 57 | 0x0000 0D72 | 2 | ECAP2 interrupt | 8 | 2 |
| INT4.3 | 58 | 0x0000 0D74 | 2 | ECAP3 interrupt | 8 | 3 |
| INT4.4 | 59 | 0x0000 0D76 | 2 | ECAP4 interrupt | 8 | 4 |
| INT4.5 | 60 | 0x0000 0D78 | 2 | ECAP5 interrupt | 8 | 5 |
| INT4.6 | 61 | 0x0000 0D7A | 2 | ECAP6 interrupt | 8 | 6 |
| INT4.7 | 62 | 0x0000 0D7C | 2 | Reserved | 8 | 7 |
| INT4.8 | 63 | 0x0000 0D7E | 2 | Reserved | 8 | 8 |
| INT4.9 | 152 | 0x0000 0E30 | 2 | Reserved | 8 | 9 |
| INT4.10 | 153 | 0x0000 0E32 | 2 | Reserved | 8 | 10 |
| INT4.11 | 154 | 0x0000 0E34 | 2 | Reserved | 8 | 11 |
| INT4.12 | 155 | 0x0000 0E36 | 2 | Reserved | 8 | 12 |
| INT4.13 | 156 | 0x0000 0E38 | 2 | Reserved | 8 | 13 |
| INT4.14 | 157 | 0x0000 0E3A | 2 | Reserved | 8 | 14 |
| INT4.15 | 158 | 0x0000 0E3C | 2 | Reserved | 8 | 15 |
| INT4.16 | 159 | 0x0000 0E3E | 2 | Reserved | 8 | 16 (Lowest) |
| **PIE Group 5 Vectors - Muxed into CPU INT5** | | | | | | |
| INT5.1 | 64 | 0x0000 0D80 | 2 | EQEP1 interrupt | 9 | 1 (Highest) |
| INT5.2 | 65 | 0x0000 0D82 | 2 | EQEP2 interrupt | 9 | 2 |
| INT5.3 | 66 | 0x0000 0D84 | 2 | EQEP3 interrupt | 9 | 3 |
| INT5.4 | 67 | 0x0000 0D86 | 2 | Reserved | 9 | 4 |
| INT5.5 | 68 | 0x0000 0D88 | 2 | CLB1 Interrupt | 9 | 5 |
| INT5.6 | 69 | 0x0000 0D8A | 2 | CLB2 Interrupt | 9 | 6 |
| INT5.7 | 70 | 0x0000 0D8C | 2 | CLB3 Interrupt | 9 | 7 |
| INT5.8 | 71 | 0x0000 0D8E | 2 | CLB4 Interrupt | 9 | 8 |
| INT5.9 | 160 | 0x0000 0E40 | 2 | SD1 interrupt | 9 | 9 |
| INT5.10 | 161 | 0x0000 0E42 | 2 | SD2 interrupt | 9 | 10 |
| INT5.11 | 162 | 0x0000 0E44 | 2 | Reserved | 9 | 11 |
| INT5.12 | 163 | 0x0000 0E46 | 2 | Reserved | 9 | 12 |
| INT5.13 | 164 | 0x0000 0E48 | 2 | Reserved | 9 | 13 |
| INT5.14 | 165 | 0x0000 0E4A | 2 | Reserved | 9 | 14 |
| INT5.15 | 166 | 0x0000 0E4C | 2 | Reserved | 9 | 15 |
| INT5.16 | 167 | 0x0000 0E4E | 2 | Reserved | 9 | 16 (Lowest) |
| **PIE Group 6 Vectors - Muxed into CPU INT6** | | | | | | |
| INT6.1 | 72 | 0x0000 0D90 | 2 | SPIA_RX interrupt | 10 | 1 (Highest) |
| INT6.2 | 73 | 0x0000 0D92 | 2 | SPIA_TX interrupt | 10 | 2 |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-4. PIE Interrupt Vectors (continued)**

| Name | Vector ID | Address | Size (x16) | Description | Core Priority | ePIE Group Priority |
|---|---|---|---|---|---|---|
| INT6.3 | 74 | 0x0000 0D94 | 2 | SPIB_RX interrupt | 10 | 3 |
| INT6.4 | 75 | 0x0000 0D96 | 2 | SPIB_TX interrupt | 10 | 4 |
| INT6.5 | 76 | 0x0000 0D98 | 2 | MCBSPA_RX interrupt | 10 | 5 |
| INT6.6 | 77 | 0x0000 0D9A | 2 | MCBSPA_TX interrupt | 10 | 6 |
| INT6.7 | 78 | 0x0000 0D9C | 2 | MCBSPB_RX interrupt | 10 | 7 |
| INT6.8 | 79 | 0x0000 0D9E | 2 | MCBSPB_TX interrupt | 10 | 8 |
| INT6.9 | 168 | 0x0000 0E50 | 2 | SPIC_RX interrupt | 10 | 9 |
| INT6.10 | 169 | 0x0000 0E52 | 2 | SPIC_TX interrupt | 10 | 10 |
| INT6.11 | 170 | 0x0000 0E54 | 2 | Reserved | 10 | 11 |
| INT6.12 | 171 | 0x0000 0E56 | 2 | Reserved | 10 | 12 |
| INT6.13 | 172 | 0x0000 0E58 | 2 | Reserved | 10 | 13 |
| INT6.14 | 173 | 0x0000 0E5A | 2 | Reserved | 10 | 14 |
| INT6.15 | 174 | 0x0000 0E5C | 2 | Reserved | 10 | 15 |
| INT6.16 | 175 | 0x0000 0E5E | 2 | Reserved | 10 | 16 (Lowest) |
| **PIE Group 7 Vectors - Muxed into CPU INT7** | | | | | | |
| INT7.1 | 80 | 0x0000 0DA0 | 2 | DMA_CH1 interrupt | 11 | 1 (Highest) |
| INT7.2 | 81 | 0x0000 0DA2 | 2 | DMA_CH2 interrupt | 11 | 2 |
| INT7.3 | 82 | 0x0000 0DA4 | 2 | DMA_CH3 interrupt | 11 | 3 |
| INT7.4 | 83 | 0x0000 0DA6 | 2 | DMA_CH4 interrupt | 11 | 4 |
| INT7.5 | 84 | 0x0000 0DA8 | 2 | DMA_CH5 interrupt | 11 | 5 |
| INT7.6 | 85 | 0x0000 0DAA | 2 | DMA_CH6 interrupt | 11 | 6 |
| INT7.7 | 86 | 0x0000 0DAC | 2 | Reserved | 11 | 7 |
| INT7.8 | 87 | 0x0000 0DAE | 2 | Reserved | 11 | 8 |
| INT7.9 | 176 | 0x0000 0E60 | 2 | Reserved | 11 | 9 |
| INT7.10 | 177 | 0x0000 0E62 | 2 | Reserved | 11 | 10 |
| INT7.11 | 178 | 0x0000 0E64 | 2 | Reserved | 11 | 11 |
| INT7.12 | 179 | 0x0000 0E66 | 2 | Reserved | 11 | 12 |
| INT7.13 | 180 | 0x0000 0E68 | 2 | Reserved | 11 | 13 |
| INT7.14 | 181 | 0x0000 0E6A | 2 | Reserved | 11 | 14 |
| INT7.15 | 182 | 0x0000 0E6C | 2 | Reserved | 11 | 15 |
| INT7.16 | 183 | 0x0000 0E6E | 2 | Reserved | 11 | 16 (Lowest) |
| **PIE Group 8 Vectors - Muxed into CPU INT8** | | | | | | |
| INT8.1 | 88 | 0x0000 0DB0 | 2 | I2CA interrupt | 12 | 1 (Highest) |
| INT8.2 | 89 | 0x0000 0DB2 | 2 | I2CA_FIFO interrupt | 12 | 2 |
| INT8.3 | 90 | 0x0000 0DB4 | 2 | I2CB interrupt | 12 | 3 |
| INT8.4 | 91 | 0x0000 0DB6 | 2 | I2CB_FIFO interrupt | 12 | 4 |
| INT8.5 | 92 | 0x0000 0DB8 | 2 | SCIC_RX interrupt | 12 | 5 |
| INT8.6 | 93 | 0x0000 0DBA | 2 | SCIC_TX interrupt | 12 | 6 |
| INT8.7 | 94 | 0x0000 0DBC | 2 | SCID_RX interrupt | 12 | 7 |
| INT8.8 | 95 | 0x0000 0DBE | 2 | SCID_TX interrupt | 12 | 8 |
| INT8.9 | 184 | 0x0000 0E70 | 2 | Reserved | 12 | 9 |
| INT8.10 | 185 | 0x0000 0E72 | 2 | Reserved | 12 | 10 |
| INT8.11 | 186 | 0x0000 0E74 | 2 | Reserved | 12 | 11 |
| INT8.12 | 187 | 0x0000 0E76 | 2 | Reserved | 12 | 12 |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-4. PIE Interrupt Vectors (continued)**

| Name | Vector ID | Address | Size (x16) | Description | Core Priority | ePIE Group Priority |
|---|---|---|---|---|---|---|
| INT8.13 | 188 | 0x0000 0E78 | 2 | Reserved | 12 | 13 |
| INT8.14 | 189 | 0x0000 0E7A | 2 | Reserved | 12 | 14 |
| INT8.15 | 190 | 0x0000 0E7C | 2 | UPPA interrupt (CPU1 only) | 12 | 15 |
| INT8.16 | 191 | 0x0000 0E7E | 2 | Reserved | 12 | 16 (Lowest) |
| **PIE Group 9 Vectors - Muxed into CPU INT9** | | | | | | |
| INT9.1 | 96 | 0x0000 0DC0 | 2 | SCIA_RX interrupt | 13 | 1 (Highest) |
| INT9.2 | 97 | 0x0000 0DC2 | 2 | SCIA_TX interrupt | 13 | 2 |
| INT9.3 | 98 | 0x0000 0DC4 | 2 | SCIB_RX interrupt | 13 | 3 |
| INT9.4 | 99 | 0x0000 0DC6 | 2 | SCIB_TX interrupt | 13 | 4 |
| INT9.5 | 100 | 0x0000 0DC8 | 2 | CANA interrupt 0 | 13 | 5 |
| INT9.6 | 101 | 0x0000 0DCA | 2 | CANA interrupt 1 | 13 | 6 |
| INT9.7 | 102 | 0x0000 0DCC | 2 | CANB interrupt 0 | 13 | 7 |
| INT9.8 | 103 | 0x0000 0DCE | 2 | CANB interrupt 1 | 13 | 8 |
| INT9.9 | 192 | 0x0000 0E80 | 2 | Reserved | 13 | 9 |
| INT9.10 | 193 | 0x0000 0E82 | 2 | Reserved | 13 | 10 |
| INT9.11 | 194 | 0x0000 0E84 | 2 | Reserved | 13 | 11 |
| INT9.12 | 195 | 0x0000 0E86 | 2 | Reserved | 13 | 12 |
| INT9.13 | 196 | 0x0000 0E88 | 2 | Reserved | 13 | 13 |
| INT9.14 | 197 | 0x0000 0E8A | 2 | Reserved | 13 | 14 |
| INT9.15 | 198 | 0x0000 0E8C | 2 | USBA interrupt (CPU1 only) | 13 | 15 |
| INT9.16 | 199 | 0x0000 0E8E | 2 | Reserved | 13 | 16 (Lowest) |
| **PIE Group 10 Vectors - Muxed into CPU INT10** | | | | | | |
| INT10.1 | 104 | 0x0000 0DD0 | 2 | ADCA_EVT interrupt | 14 | 1 (Highest) |
| INT10.2 | 105 | 0x0000 0DD2 | 2 | ADCA2 interrupt | 14 | 2 |
| INT10.3 | 106 | 0x0000 0DD4 | 2 | ADCA3 interrupt | 14 | 3 |
| INT10.4 | 107 | 0x0000 0DD6 | 2 | ADCA4 interrupt | 14 | 4 |
| INT10.5 | 108 | 0x0000 0DD8 | 2 | ADCB_EVT interrupt | 14 | 5 |
| INT10.6 | 109 | 0x0000 0DDA | 2 | ADCB2 interrupt | 14 | 6 |
| INT10.7 | 110 | 0x0000 0DDC | 2 | ADCB3 interrupt | 14 | 7 |
| INT10.8 | 111 | 0x0000 0DDE | 2 | ADCB4 interrupt | 14 | 8 |
| INT10.9 | 200 | 0x0000 0E90 | 2 | ADCC_EVT interrupt | 14 | 9 |
| INT10.10 | 201 | 0x0000 0E92 | 2 | ADCC2 interrupt | 14 | 10 |
| INT10.11 | 202 | 0x0000 0E94 | 2 | ADCC3 interrupt | 14 | 11 |
| INT10.12 | 203 | 0x0000 0E96 | 2 | ADCC4 interrupt | 14 | 12 |
| INT10.13 | 204 | 0x0000 0E98 | 2 | ADCD_EVT interrupt | 14 | 13 |
| INT10.14 | 205 | 0x0000 0E9A | 2 | ADCD2 interrupt | 14 | 14 |
| INT10.15 | 206 | 0x0000 0E9C | 2 | ADCD3 interrupt | 14 | 15 |
| INT10.16 | 207 | 0x0000 0E9E | 2 | ADCD4 interrupt | 14 | 16 (Lowest) |
| **PIE Group 11 Vectors - Muxed into CPU INT11** | | | | | | |
| INT11.1 | 112 | 0x0000 0DE0 | 2 | CLA1_1 interrupt | 15 | 1 (Highest) |
| INT11.2 | 113 | 0x0000 0DE2 | 2 | CLA1_2 interrupt | 15 | 2 |
| INT11.3 | 114 | 0x0000 0DE4 | 2 | CLA1_3 interrupt | 15 | 3 |
| INT11.4 | 115 | 0x0000 0DE6 | 2 | CLA1_4 interrupt | 15 | 4 |
| INT11.5 | 116 | 0x0000 0DE8 | 2 | CLA1_5 interrupt | 15 | 5 |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-4. PIE Interrupt Vectors (continued)**

| Name | Vector ID | Address | Size (x16) | Description | Core Priority | ePIE Group Priority |
|---|---|---|---|---|---|---|
| INT11.6 | 117 | 0x0000 0DEA | 2 | CLA1_6 interrupt | 15 | 6 |
| INT11.7 | 118 | 0x0000 0DEC | 2 | CLA1_7 interrupt | 15 | 7 |
| INT11.8 | 119 | 0x0000 0DEE | 2 | CLA1_8 interrupt | 15 | 8 |
| INT11.9 | 208 | 0x0000 0EA0 | 2 | Reserved | 15 | 9 |
| INT11.10 | 209 | 0x0000 0EA2 | 2 | Reserved | 15 | 10 |
| INT11.11 | 210 | 0x0000 0EA4 | 2 | Reserved | 15 | 11 |
| INT11.12 | 211 | 0x0000 0EA6 | 2 | Reserved | 15 | 12 |
| INT11.13 | 212 | 0x0000 0EA8 | 2 | Reserved | 15 | 13 |
| INT11.14 | 213 | 0x0000 0EAA | 2 | Reserved | 15 | 14 |
| INT11.15 | 214 | 0x0000 0EAC | 2 | Reserved | 15 | 15 |
| INT11.16 | 215 | 0x0000 0EAE | 2 | Reserved | 15 | 16 (Lowest) |
| **PIE Group 12 Vectors - Muxed into CPU INT12** | | | | | | |
| INT12.1 | 120 | 0x0000 0DF0 | 2 | XINT3 interrupt | 16 | 1 (Highest) |
| INT12.2 | 121 | 0x0000 0DF2 | 2 | XINT4 interrupt | 16 | 2 |
| INT12.3 | 122 | 0x0000 0DF4 | 2 | XINT5 interrupt | 16 | 3 |
| INT12.4 | 123 | 0x0000 0DF6 | 2 | Reserved | 16 | 4 |
| INT12.5 | 124 | 0x0000 0DF8 | 2 | Reserved | 16 | 5 |
| INT12.6 | 125 | 0x0000 0DFA | 2 | VCU interrupt | 16 | 6 |
| INT12.7 | 126 | 0x0000 0DFC | 2 | FPU_OVERFLOW interrupt | 16 | 7 |
| INT12.8 | 127 | 0x0000 0DFE | 2 | FPU_UNDERFLOW interrupt | 16 | 8 |
| INT12.9 | 216 | 0x0000 0EB0 | 2 | EMIF_ERROR interrupt | 16 | 9 |
| INT12.10 | 217 | 0x0000 0EB2 | 2 | RAM_CORRECTABLE_ERROR interrupt | 16 | 10 |
| INT12.11 | 218 | 0x0000 0EB4 | 2 | FLASH_CORRECTABLE_ERROR interrupt | 16 | 11 |
| INT12.12 | 219 | 0x0000 0EB6 | 2 | RAM_ACCESS_VIOLATION interrupt | 16 | 12 |
| INT12.13 | 220 | 0x0000 0EB8 | 2 | SYS_PLL_SLIP interrupt | 16 | 13 |
| INT12.14 | 221 | 0x0000 0EBA | 2 | AUX_PLL_SLIP interrupt | 16 | 14 |
| INT12.15 | 222 | 0x0000 0EBC | 2 | CLA_OVERFLOW interrupt | 16 | 15 |
| INT12.16 | 223 | 0x0000 0EBE | 2 | CLA_UNDERFLOW interrupt | 16 | 16 (Lowest) |

## 3.5 Exceptions and Non-Maskable Interrupts

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

### *3.5.1 Configuring and Using NMIs*

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

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



Copyright © 2024 Texas Instruments Incorporated

### 3.5.2 Emulation Considerations

The NMI watchdog counter behaves as follows under debug conditions:

| | |
|---|---|
| CPU Suspended | When the CPU is suspended, the NMI watchdog counter 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. |

### 3.5.3 NMI Sources

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

#### 3.5.3.1 Missing Clock Detection

The missing clock detection logic monitors OSCCLK for failure. If the OSCCLK source stops, the PLL is bypassed, OSCCLK is connected to INTOSC1, and NMIs are fired to both CPUs . For more information on missing clock detection, see Section 3.6.2.

#### 3.5.3.2 RAM Uncorrectable ECC Error

A single-bit parity error, double-bit ECC data error, or single-bit ECC address error in a RAM read triggers an NMI. This applies to CPU, CLA, and DMA reads. Single-bit ECC data errors do not trigger an NMI, but can optionally trigger a normal peripheral interrupt. For more information on RAM error detection, see Section 3.11.1.8.

#### 3.5.3.3 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. For more information on Flash error detection, see Section 3.12.10.

#### 3.5.3.4 NMI Vector Fetch Mismatch

Each CPU's Peripheral Interrupt Expansion module (PIE) has redundant vector tables. If a mismatch in these tables is detected during a vector fetch, a user-specified error handler is run instead of the ISR. If the vector fetch was caused by an NMI, a second NMI is fired to the other CPU. Mismatches for other interrupts do not trigger an NMI. For more information about the vector address check, see Section 3.6.4.

#### 3.5.3.5 CPU2 Watchdog or NMI Watchdog Reset

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

Watchdog interrupts do not trigger an NMI.

### 3.5.4 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 in the *TMS320C28x DSP CPU and Instruction Set Reference Guide*.



Copyright © 2024 Texas Instruments Incorporated

**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 Safety Features

This section gives details on features that monitor device operation during run-time to detect any error in operation.

### *3.6.1 Write Protection on Registers*

#### 3.6.1.1 LOCK Protection on System Configuration Registers

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

#### 3.6.1.2 EALLOW Protection

Several control registers are protected from spurious CPU writes by the EALLOW protection mechanism. The EALLOW bit in status register 1 (ST1) indicates the state of protection as shown in Table 3-5.

**Table 3-5. Access to EALLOW-Protected Registers**

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

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

At reset, the EALLOW bit is cleared, enabling EALLOW protection. While protected, all writes to protected registers by the CPU are ignored and only CPU reads, JTAG reads, and JTAG writes are allowed. If this bit is set, by executing the EALLOW instruction, the CPU is allowed to write freely to protected registers. After modifying registers, the registers can once again be protected by executing the EDIS instruction to clear the EALLOW bit.

### *3.6.2 Missing Clock Detection Logic*

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

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

1. The primary clock (OSCCLK) clock keeps ticking a 7-bit counter (named as MCDPCNT). This counter is asynchronously reset with $\overline{\text{XRS}}$.
2. The secondary clock (INTOSC1) clock keeps ticking a 13-bit counter (named as MCDSCNT). This counter is asynchronously reset with $\overline{\text{XRS}}$.
3. Each time MCDPCNT overflows, the MCDSCNT counter is reset. Thus, if OSCCLK is present or is not slower than INTOSC1 by a factor of 64, MCDSCNT never overflows.
4. 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.
5. The above check is continuously active, unless the MCD is disabled using MCDCR register (by making the MCLKOFF bit 1)
6. If the circuit ever detects a missing OSCCLK, the following occurs:
   - The MCLKSTS flag is set.
   - The MCDSCNT counter is frozen to prevent further missing clock detection.
   - The CLOCKFAIL signal goes high, which generates TRIP events to PWM modules and fires NMIs to CPU1.NMIWD and CPU2.NMIWD.



Copyright © 2024 Texas Instruments Incorporated

- PLL is forcefully bypassed and OSCCLK is switched to INTOSC1 (after the PLLSYSCLK divider). PLLMULT is zeroed out automatically in this case.
- While the MCLKSTS bit is set, the OSCCLKSRCSEL bits have no effect and OSCCLK is forcefully connected to INTOSC1.
- PLLRAWCLK going to the system is switched to INTOSC1 automatically.

7. If the MCLKCLR bit is written (W = 1 bit), MCLKSTS bit are 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. To lock the PLL after a missing clock detection, switch the clock source to INTOSC1 (using OSCCLKSRCSEL register), do a MCLKCLR, and re-lock the PLL.
8. The MCD is enabled at power up. There is no support for a missing clock detection, if INTOSC2 is failed from the device power-up.

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

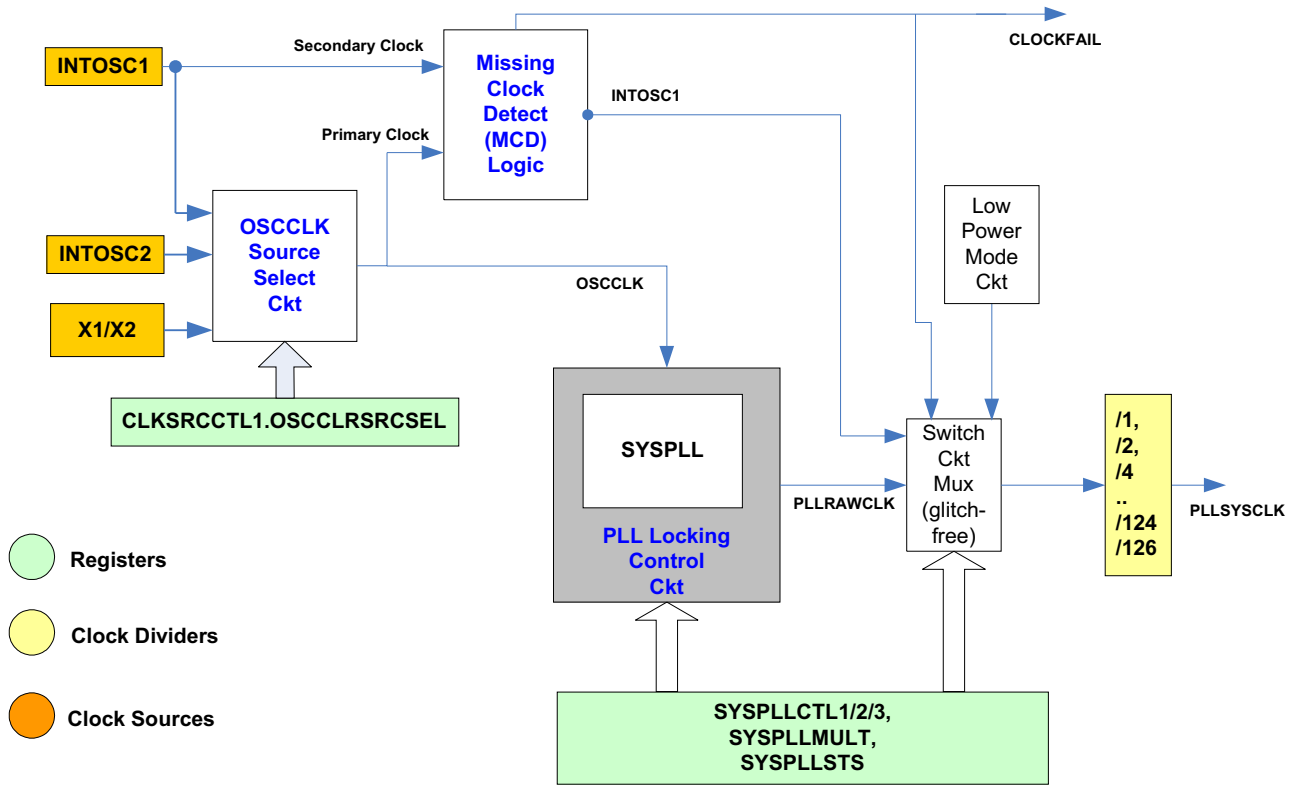



**Figure 3-3. Missing Clock Detection Logic**

**Note**

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

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

### *3.6.3 PLLSLIP Detection*

The PLL SLIP detection on this device can detect if the PLL reference clock goes too high or too slow while PLL is locked. An interrupt to both the CPUs is triggered as shown in the ePIE table in Section 3.4. Apart from the interrupt to both the CPUs, the PLLSTS.SLIP bit is set for user software to check the error.

The SLIP detection is available on both SYSPLL and AUXPLL.



Copyright © 2024 Texas Instruments Incorporated

### 3.6.4 CPU1 and CPU2 PIE Vector Address Validity Check

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

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

Following is the behavior of accesses to the ePIE memories:

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

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

1. If the PIEVERRADDR register (default value 0x3F FFFF) is not initialized, the default error handler at address 0x3FFFBE gets executed. But, when the PIEVERRADDR register is initialized to the address of the user-defined routine, the user-defined routine is executed instead of the default error handler.
   **Note:** Each CPU has a copy of the PIE Vector Fetch Error Handler register (CPU1.PIEVERRADDR and CPU2.PIEVERRADDR).
2. Hardware also generates EPWM Trip signals that trips the PWM outputs using TRIPIN15.
3. An NMI to the other CPU is sent, if the current mismatch is during a vector fetch. For example, on an NMI vector fetch error for CPU2 an NMI is also fired to CPU1.NMIWD.

If there is no mismatch, the correct vector is jammed onto the C28x program control.

### 3.6.5 NMIWDs

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

### 3.6.6 ECC and Parity Enabled RAMs, Shared RAMs Protection

Each CPU subsystem has different RAM blocks. Few RAM blocks are ECC-enabled and others are parity-enabled. All single-bit errors in ECC RAM are auto-corrected and an error counter is incremented every time a single bit error is detected. If the error counter reaches a predefined user configured limit, an interrupt is generated to the corresponding CPU. A typical threshold setting to avoid triggering on transient errors which are corrected, but identify persistent faults is 10. Refer to Section 3.11 for more details on RAM errors.

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

### 3.6.7 ECC Enabled Flash Memory

When ECC is programmed and enabled, Flash single-bit errors are corrected automatically by ECC logic before giving data to the CPU, but the errors are not corrected in Flash memory. Flash memory still contains wrong data until another erase/program operation happens to correct the Flash contents. Irrespective of whether the error interrupt is enabled or disabled, single-bit errors are always corrected before giving data to the CPU. When the interrupt is disabled, users can check the single-bit error counter register for any single-bit error occurrences. The error counter stops incrementing once the value is equal to the threshold+1. It is always suggested to set the threshold register to a non-zero value so that the error counter can increment. It is up to the user to decide the threshold value at which to reprogram the Flash with the correct data. A typical threshold setting to avoid triggering on transient errors that are corrected, but identify persistent faults is 10. When ECC is programmed and enabled, Flash uncorrectable errors end up triggering an NMI to the respective CPU. Please refer to Section 3.11 for more details on Flash error correction and error catching mechanisms.



Copyright © 2024 Texas Instruments Incorporated

### 3.6.8 ERRORSTS Pin

The ERRORSTS pin is an ‘always output’ pin and remains low until an error is detected inside the chip. On an error, the ERRORSTS pin goes high 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-high pin, users who care about the state of this pin during power-up must connect an external pull-down on this pin.

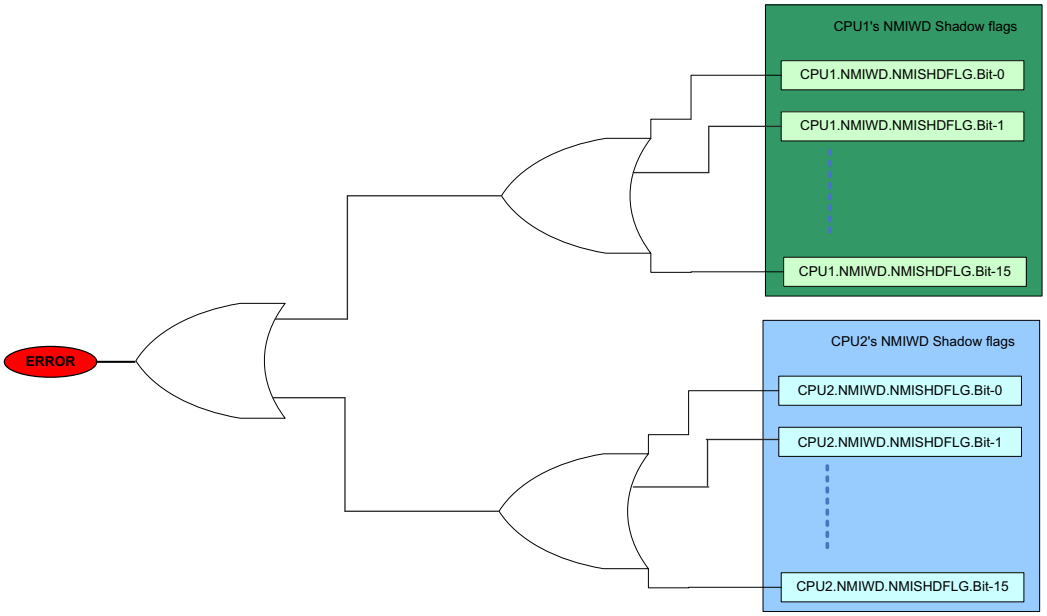



**Figure 3-4. ERRORSTS Pin Diagram**

## 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 provides an overview of the device clocking system.

**Note**

While the CLK_CFG_REGS registers are mapped to both CPU1 and CPU2, there are not unique settings per CPU, and the impact of these registers is global. Only one CPU can read/write to these registers at a time; this is controlled by the Clock Control Semaphore Register (CLKSEM) in the CLK_CFG_REGs. This register is unique in that either CPU can read and write to the register.

The default/2 divider for ePWMs and EMIFs is not shown in Figure 3-5.



Copyright © 2024 Texas Instruments Incorporated

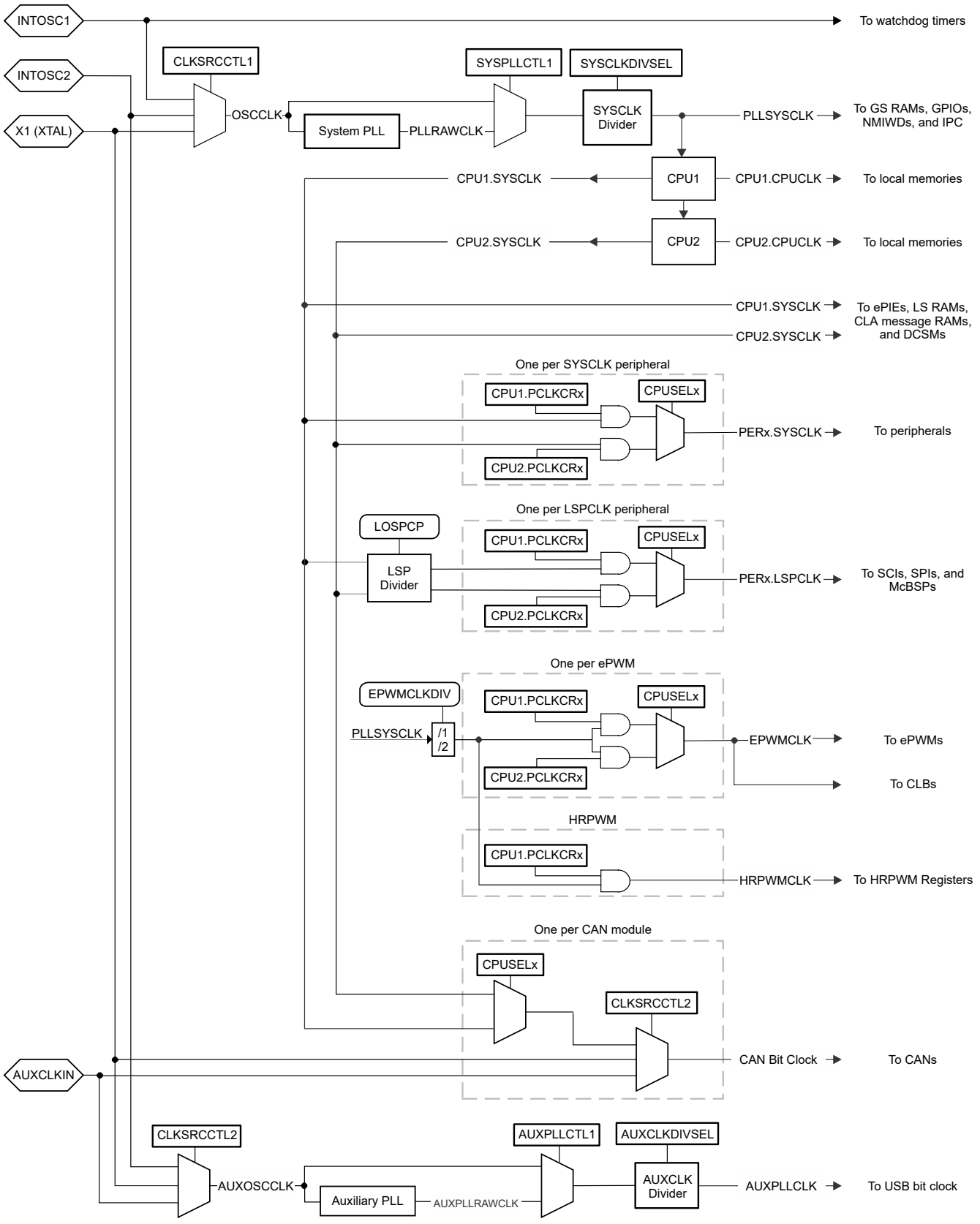



**Figure 3-5. Clocking System**



Copyright © 2024 Texas Instruments Incorporated

### 3.7.1 Clock Sources

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

#### 3.7.1.1 Primary Internal Oscillator (INTOSC2)

At power-up, the device is clocked from an on-chip 10 MHz oscillator (INTOSC2). INTOSC2 is the primary internal clock source, and is the default system clock at reset. It is used to run the boot ROM and can be used as the system clock source for the application. Note that INTOSC2's frequency tolerance is too loose to meet the timing requirements for CAN and USB, so an external clock must be used to support those features.

#### 3.7.1.2 Backup Internal Oscillator (INTOSC1)

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

#### 3.7.1.3 External Oscillator (XTAL)

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

- A single-ended 3.3V external clock. The clock signal must be connected to X1 while X2 is left unconnected, as shown in Figure 3-6.

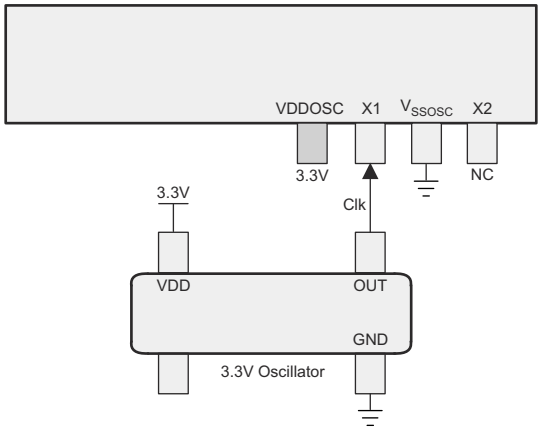



**Figure 3-6. Single-ended 3.3V External Clock**

- An external crystal. The crystal must be connected across X1 and X2 with the load capacitors connected to VSSOSC as shown in Figure 3-7.

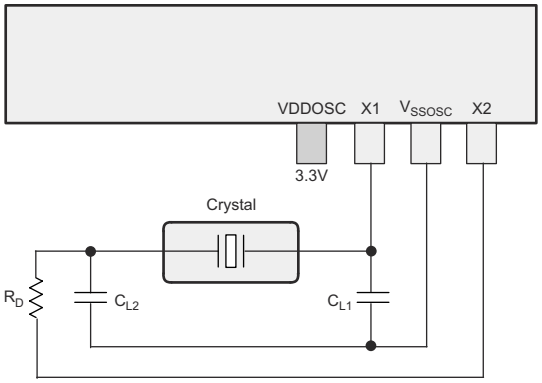



**Figure 3-7. External Crystal**



Copyright © 2024 Texas Instruments Incorporated

- An external resonator. The resonator must be connected across X1 and X2 with the ground connected to VSSOSC as shown in Figure 3-8.

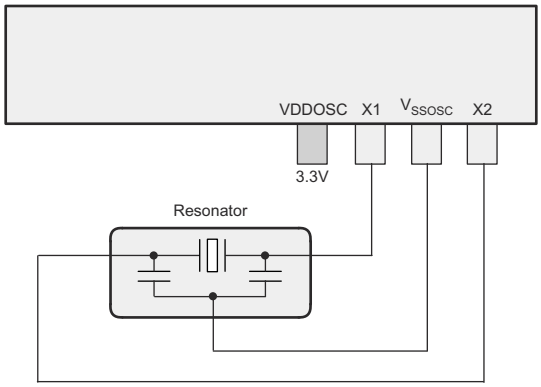



**Figure 3-8. External Resonator**

**Note**

All 3 external clocking modes require the XTALOFF bit in CLKSRCCTL1 register to be set to 0. This enables the path and circuitry required for the clock to propagate to the device.

#### 3.7.1.4 Auxiliary Clock Input (AUXCLKIN)

An additional external clock source is supported on GPIO133 (AUXCLKIN). This must be a single-ended 3.3V external clock and can be used as the source for the USB and CAN bit clocks. Frequency limits and timing requirements can be found in the device data sheet. The external clock must be connected directly to the GPIO133 pin, as shown in Figure 3-9.

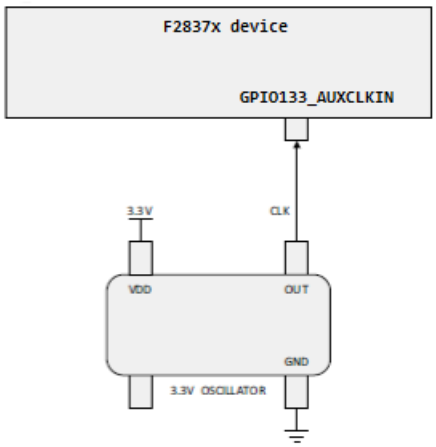



**Figure 3-9. AUXCLKIN**

### *3.7.2 Derived Clocks*

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

#### 3.7.2.1 Oscillator Clock (OSCCLK)

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



Copyright © 2024 Texas Instruments Incorporated

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

The system PLL allows the device to run at its maximum rated operating frequency, and in most applications generates the main system clock. This PLL uses OSCCLK as a reference, and features a fractional multiplier and slip detection. For configuration instructions, see Section 3.7.6.

#### 3.7.2.3 Auxiliary Oscillator Clock (AUXOSCCLK)

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

#### 3.7.2.4 Auxiliary PLL Output Clock (AUXPLLRAWCLK)

The auxiliary PLL is used to generate a 60 MHz clock for the USB module. This PLL uses AUXOSCCLK as a reference, and features a fractional multiplier and slip detection. For configuration instructions, see Section 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 system control registers, GS RAMs, IPC module, GPIO qualification, and NMI watchdog timers have their own clock domain (PLLSYSCLK). Despite the name, PLLSYSCLK may be connected to the system PLL (PLLRAWCLK) or to OSCCLK. The chosen clock source is run through a frequency divider, which is configured via the SYSCLKDIVSEL register. PLLSYSCLK is gated in HALT mode.

#### 3.7.3.2 CPU Clock (CPUCLK)

Each CPU has a clock (CPU1.CPUCLK and CPU2.CPUCLK) that is used to clock the CPU, the coprocessors, the private RAMs (M0, M1, D0, and D1), and the boot ROM 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)

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

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

A peripheral may be assigned to either CPU. That is, code for a peripheral can be executed from either CPU1 or CPU2. CPUSELx register is used assign a peripheral to either CPU1 or CPU2. This register must be configured prior to enabling the clock for the chosen peripheral since the clock for each peripheral is derived from the selected CPU subsystem. The clock multiplexer controlled by the CPUSELx register is not glitch-free. Therefore the CPUSELx register must be configured before the PCLKCRx register. Note that the reset for each peripheral is also driven from the selected CPU.

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

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



Copyright © 2024 Texas Instruments Incorporated

#### 3.7.3.5 USB Auxiliary Clock (AUXPLLCLK)

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

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

#### 3.7.3.6 CAN Bit Clock

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

#### 3.7.3.7 CPU Timer2 Clock (TIMER2CLK)

CPU timers 0 and 1 are connected to PERx.SYSCLK. Timer 2 is connected to PERx.SYSCLK by default, but may also be connected to INTOSC1, INTOSC2, XTAL, or AUXPLLCLK via the TMR2CLKCTL register. This register also provides a separate prescale divider for timer 2. If a source other than SYSCLK is used, the SYSCLK frequency must be at least twice the source frequency to ensure correct sampling. Each CPU has its own independent CPU timers and TMR2CLKCTL register.

The main reason to use a non-SYSCLK source would be for internal frequency measurement. In most applications, timer 2 runs off of the 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, GPIO73. The available clock sources are PLLSYSCLK, PLLRAWCLK, CPU1.SYSCLK, CPU2.SYSCLK, AUXPLLRAWCLK, INTOSC1, and INTOSC2.

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



Copyright © 2024 Texas Instruments Incorporated

### 3.7.5 Clock Connectivity

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

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

| Clock Domain | CPU1 Subsystem | CPU2 Subsystem | Shared Modules |
|---|---|---|---|
| CPUx.CPUCLK | CPU1<br>CPU1.VCU<br>CPU1.FPU<br>CPU1.TMU<br>CPU1.M0 - M1 RAMs<br>CPU1.D0 - D1 RAMs<br>CPU1.BootROM<br>CPU1.Flash | CPU2<br>CPU2.VCU<br>CPU2.FPU<br>CPU2.TMU<br>CPU2.M0 - M1 RAMs<br>CPU2.D0 - D1 RAMs<br>CPU2.BootROM<br>CPU2.Flash | |
| CPUx.SYSCLK | CPU1.ePIE<br>CPU1.LS0 - LS5 RAMs<br>CPU1.CLA1 Message RAMs<br>CPU1.DCSM | CPU2.ePIE<br>CPU2.LS0 - LS5 RAMs<br>CPU2.CLA1 Message RAMs<br>CPU2.DCSM | |
| PLLSYSCLK | CPU1.NMIWD<br>EMIF1 | CPU2.NMIWD | GS0 - GS15 RAMs<br>GPIO Input Sync and Qual<br>IPC |
| PERx.SYSCLK | CPU1.CLA1<br>CPU1.DMA<br>CPU1.Timer0 - 2<br>EMIF2<br>uPP A | CPU2.CLA1<br>CPU2.DMA<br>CPU2.Timer0 - 2 | ADCA - D<br>CLB1 - 4<br>CMPSS1 - 8<br>DACA - C<br>ePWM1 - 12<br>eCAP1 - 6<br>eQEP1 - 3<br>I2CA - B<br>McBSPA - B<br>SDFM1 - 8 |
| PERx.LSPCLK | | | McBSPA - B<br>SCIA - D<br>SPIA - C |
| CAN Bit Clock | | | CANA - B |
| AUXPLLCLK | USB | | |
| WDCLK (INTOSC1) | CPU1.Watchdog | CPU2.Watchdog | |



Copyright © 2024 Texas Instruments Incorporated

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

| Module Name | Clock Domain |
|---|---|
| ADCA - D | PERx.SYSCLK |
| Boot ROM | CPUx.CPUCLK |
| CANA - B | CAN Bit Clock |
| CLA | PERx.SYSCLK |
| CLA Message RAMs | CPUx.SYSCLK |
| CLB1 - 4 | PERx.SYSCLK |
| CMPSS1 - 8 | PERx.SYSCLK |
| CPU | CPUx.CPUCLK |
| CPU Timers | PERx.SYSCLK |
| D0 - D1 RAMs | CPUx.CPUCLK |
| DACA - C | PERx.SYSCLK |
| DCSM | CPUx.SYSCLK |
| DMA | PERx.SYSCLK |
| eCAP1 - 6 | PERx.SYSCLK |
| EMIF1 | PLLSYSCLK |
| EMIF2 | PERx.SYSCLK |
| ePIE | CPUx.SYSCLK |
| ePWM | PERx.SYSCLK |
| eQEP1 - 3 | PERx.SYSCLK |
| Flash | CPUx.CPUCLK |
| FPU | CPUx.CPUCLK |
| GS0 - GS15 RAMs | PLLSYSCLK |
| I2CA - B | PERx.SYSCLK |
| IPC | PLLSYSCLK |
| LS0 - LS5 RAMs | CPUx.SYSCLK |
| M0 - M1 RAMs | CPUx.CPUCLK |
| McBSPA - B | PERx.LSPCLK |
| NMIWD | PLLSYSCLK |
| SCIA - D | PERx.LSPCLK |
| SDFM1 - 8 | PERx.SYSCLK |
| SPIA - C | PERx.LSPCLK |
| TMU | CPUx.CPUCLK |
| uPP | PERx.SYSCLK |
| USB | AUXPLLCLK |
| VCU | CPUx.CPUCLK |
| Watchdog Timer | WDCLK (INTOSC1) |

### *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. The concerns must provide answers to the following questions:

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



Copyright © 2024 Texas Instruments Incorporated

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

#### 3.7.6.1 Choosing PLL Settings

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

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

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

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

Many combinations of multiplier and divider can produce the same output frequency. However, the product of the reference clock frequency and the multiplier (known as the VCO frequency) must be in the range specified in the data sheet.

**Note**

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

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

#### 3.7.6.2 System Clock Setup

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

1. Select the reference clock source (OSCCLK) by writing to CLKSRCCTL1.OSCCLKSRCSEL.
2. Set up the system PLL: (see the InitSysPll() function in your devices controlSUITE installation for an example):
   a. Bypass the PLL by clearing SYSPLLCTL1[PLLCLKEN].
   b. Set the system clock divider to /1 to make sure of the fastest PLL configuration by clearing SYSCLKDIVSEL[ PLLSYSCLKDIV].
   c. Set the integral and fractional multipliers by simultaneously writing them both to SYSPLLMULT. This automatically enables the PLL. Be sure that the product of OSCCLK and the multiplier is in the range specified in the data sheet.
   d. Lock the PLL five times (see your device errata for details). This number can be increased depending on application requirements. A higher number of lock attempts helps to make sure of a successful PLL start.
   e. Set the system clock divider one setting higher than the final desired value. For example ClkCfgRegs.SYSCLKDIVSEL.bit.PLLSYSCLKDIV = divsel + 1. This limits the current increase when switching to the PLL.
   f. Set up the watchdog to reset the device. Note that the SCRS[WDOVERRIDE] bit must not be cleared prior to locking the PLL.
   g. Set the SYSDBGCTL[BIT_0] bit. This bit is only reset by a POR reset. If the watchdog has to reset the device due to an issue with switching to the PLL, this bit can be checked in the reset handler to determine the reset was caused by a PLL error.
   h. Switch to the PLL as the system clock by setting SYSPLLCTL1[PLLCLKEN].
   i. Clear the SYSDBGCTL[BIT_0] bit.
   j. Change the divider to the appropriate value.
   k. Reconfigure the watchdog as needed for the application.



Copyright © 2024 Texas Instruments Incorporated

**Note**

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

#### 3.7.6.3 USB Auxiliary Clock Setup

See the InitAuxPll() function in your device's controlSUITE installation for an example.

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

1. Select the reference clock source (AUXOSCCLK) by writing to CLKSRCCTL2.AUXOSCCLKSRCSEL.
2. Wait two AUXOSCCLK cycles.
3. Set up the auxiliary PLL. If the PLL is not needed, bypass the PLL and power the PLL down by writing a 0 to AUXPLLCTL1.PLLEN. To use the PLL:
   a. Set the desired auxiliary clock divider by writing to AUXCLKDIVSEL.AUXPLLDIV.
   b. ) Configure CPU Timer 2 to be clocked from AUXPLL. Keep the counter frozen.
   c. ) Power down the AUXPLL by clearing AUXPLLCTL1[PLLEN].
   d. Set the integral and fractional multipliers simultaneously. This automatically enables the PLL. Be sure that the product of AUXOSCCLK and the multiplier is in the range specified in the data sheet.
   e. Wait for the PLL to lock by polling the AUXPLLSTS.LOCKS bit. This takes 16 µs plus 1024 AUXOSCCLK cycles.
   f. Connect the auxiliary PLL output clock (AUXPLLRAWCLK) to AUXPLLCLK by writing a 1 to AUXPLLCTL1.PLLCLKEN.
   g. Start CPU Timer 2. In a large for() loop, continue polling the TCR[TIF] overflow flag. If the flag is set, the AUXPLL started correctly. If not set, repeat steps (c) through (g). The auxiliary clock configuration can be changed at run time. Changing the AUXOSCCLK source automatically bypasses the PLL and sets the multiplier to zero. Changing the multiplier from one non-zero value to another temporarily bypasses the PLL until the PLL re-locks.

The auxiliary clock configuration can be changed at run time. Changing the AUXOSCCLK source automatically bypasses the PLL and sets the multiplier to zero. Changing the multiplier from one non-zero value to another temporarily bypasses the PLL until the PLL re-locks.

**Note**

If the AUXOSCCLK source is changed on the same AUXOSCCLK cycle as the multiplier, the PLL is disabled but the AUXPLLMULT register shows the written value. This can happen when the system PLL is enabled before configuring the auxiliary PLL (CPUCLK >> AUXOSCCLK). To avoid this issue, wait 2 AUXOSCCLK cycles between changing the clock source and writing to AUXPLLMULT.



Copyright © 2024 Texas Instruments Incorporated

#### 3.7.6.4 Clock Configuration Examples

**Example 1:** Using a crystal (15 MHz) as a reference, generates a CPU frequency of 100 MHz and a USB clock of 60 MHz:

CLKSRCCTL1.OSCCLKSRCSEL = 0x1

SYSPLLMULT.IMULT = 26 (0x1A)

SYSPLLMULT.FMULT = .50 (0x2)

SYSCLKDIVSEL.PLLSYSCLKDIV = 4 (0x2)

SYSPLLCTL1.PLLCLKEN = 1

PERCLKDIVSEL.EPWMCLKDIV = 1 (0x0)

PERCLKDIVSEL.EMIF1CLKDIV = 1 (0x0)

PERCLKDIVSEL.EMIF2CLKDIV = 1 (0x0)

CLKSRCCTL2.AUXOSCCLKSRCSEL = 0x1

AUXPLLMULT.IMULT = 8 (0x08)

AUXPLLMULT.FMULT = .00 (0x0)

AUXCLKDIVSEL.AUXPLLDIV = 2 (0x1)

AUXPLLCTL1.PLLCLKEN = 1

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

**Example 2:** Using INTOSC2 (10 MHz) as a reference, generates a CPU frequency of 200 MHz - 3%:

CLKSRCCTL1.OSCCLKSRCSEL = 0x0

SYSPLLMULT.IMULT = 38 (0x26)

SYSPLLMULT.FMULT = .75 (0x3)

SYSCLKDIVSEL.PLLSYSCLKDIV = 2 (0x1)

SYSPLLCTL1.PLLCLKEN = 1

### *3.7.7 Clock (OSCCLK) Failure Detection*

To achieve safety diagnostic, Missing Clock Detection (MCD) can be used. Table 3-8 lists the details.

**Table 3-8. 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. |

#### 3.7.7.1 Missing Clock Detection Logic

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

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



Copyright © 2024 Texas Instruments Incorporated

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

Figure 3-10 shows the missing clock logic functional flow.

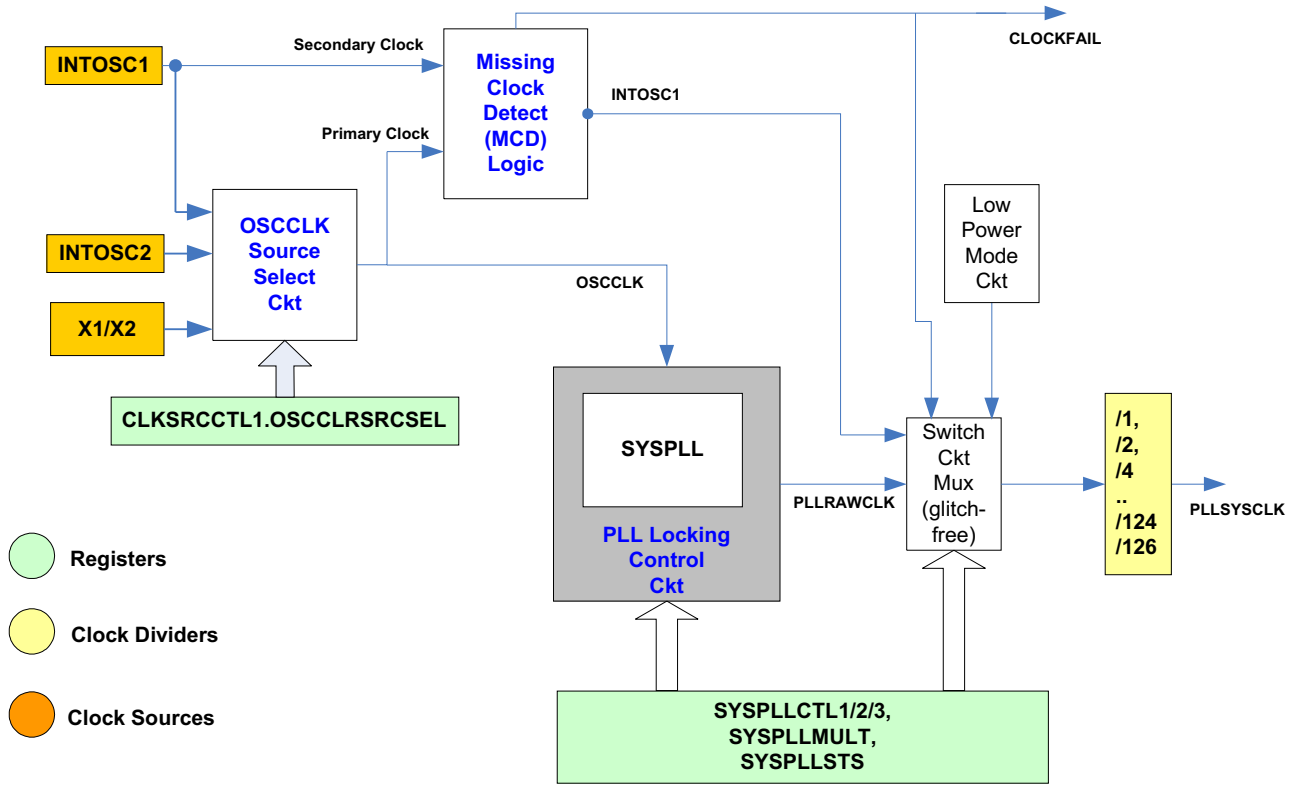



**Figure 3-10. Missing Clock Detection Logic**



Copyright © 2024 Texas Instruments Incorporated

**Note**

On a complete clock failure when OSCCLK is dead, it can take a maximum time of 8192 INTOSC1 cycles (that is, 0.8192 ms) 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 3-11.

CPU-Timer0 and CPU-Timer1 can be used in user applications. CPU-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 CPU-Timer2 can be used in the application. CPU-Timer0 and CPU-Timer1 run off of SYSCLK. CPU-Timer2 normally runs off of SYSCLK, but can also use INTOSC1, INTOSC2, XTAL, and AUXPLLCLK. The CPU-Timer interrupt signals (TINT0, TINT1, TINT2) are connected as shown in Figure 3-12.

**Note**

If a source other than SYSCLK is used for CPU-Timer2, the SYSCLK frequency must be at least twice the source frequency to make sure of correct sampling.

The CPU-Timer2 pre-scaler is implemented as a post-scale of the results

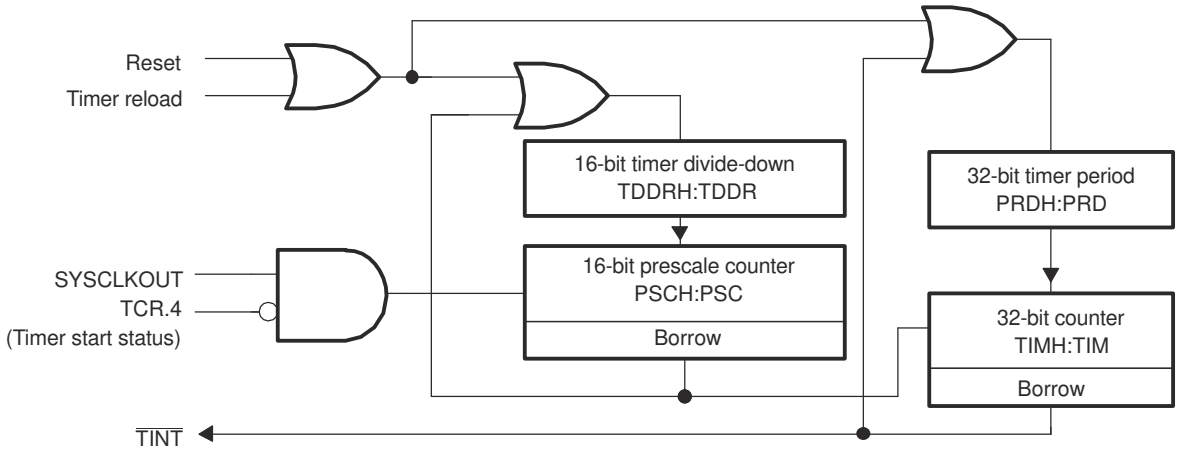
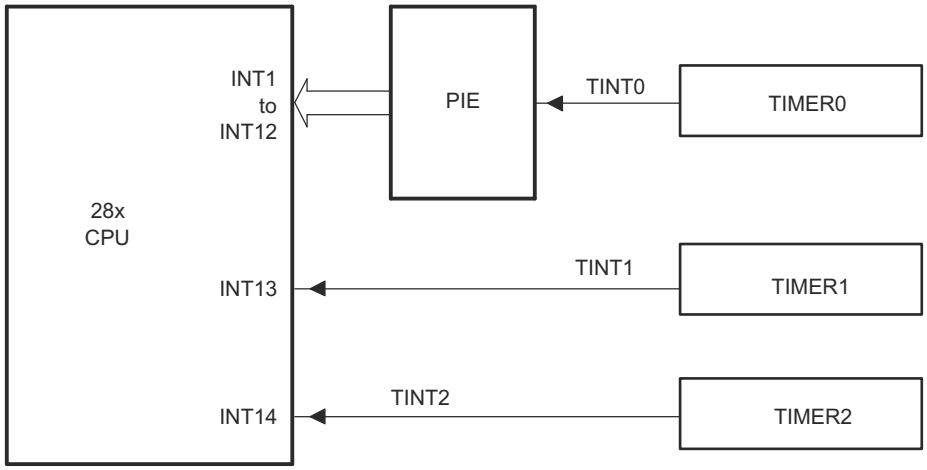



**Figure 3-11. CPU-Timers**



Copyright © 2024 Texas Instruments Incorporated




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

**Figure 3-12. CPU-Timer Interrupts Signals and Output Signal**

The general operation of the 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[TDDRH:TDDR]+1) SYSCLKOUT cycles, where TDDRH:TDDR is the timer divider.
- When the counter reaches 0, a timer interrupt output signal generates an interrupt pulse.

The registers listed in Section 3.17 are used to configure the timers.



Copyright © 2024 Texas Instruments Incorporated

## 3.9 Watchdog Timers

The watchdog module generates an output pulse 512 watchdog clocks (WDCLKs) wide whenever the 8-bit watchdog up counter has reached the maximum value. The watchdog clock source is INTOSC1. Software must periodically write a 0x55 + 0xAA sequence into the watchdog key register to reset the watchdog counter. The counter can also be disabled. Figure 3-13 shows the various functional blocks within the watchdog module.

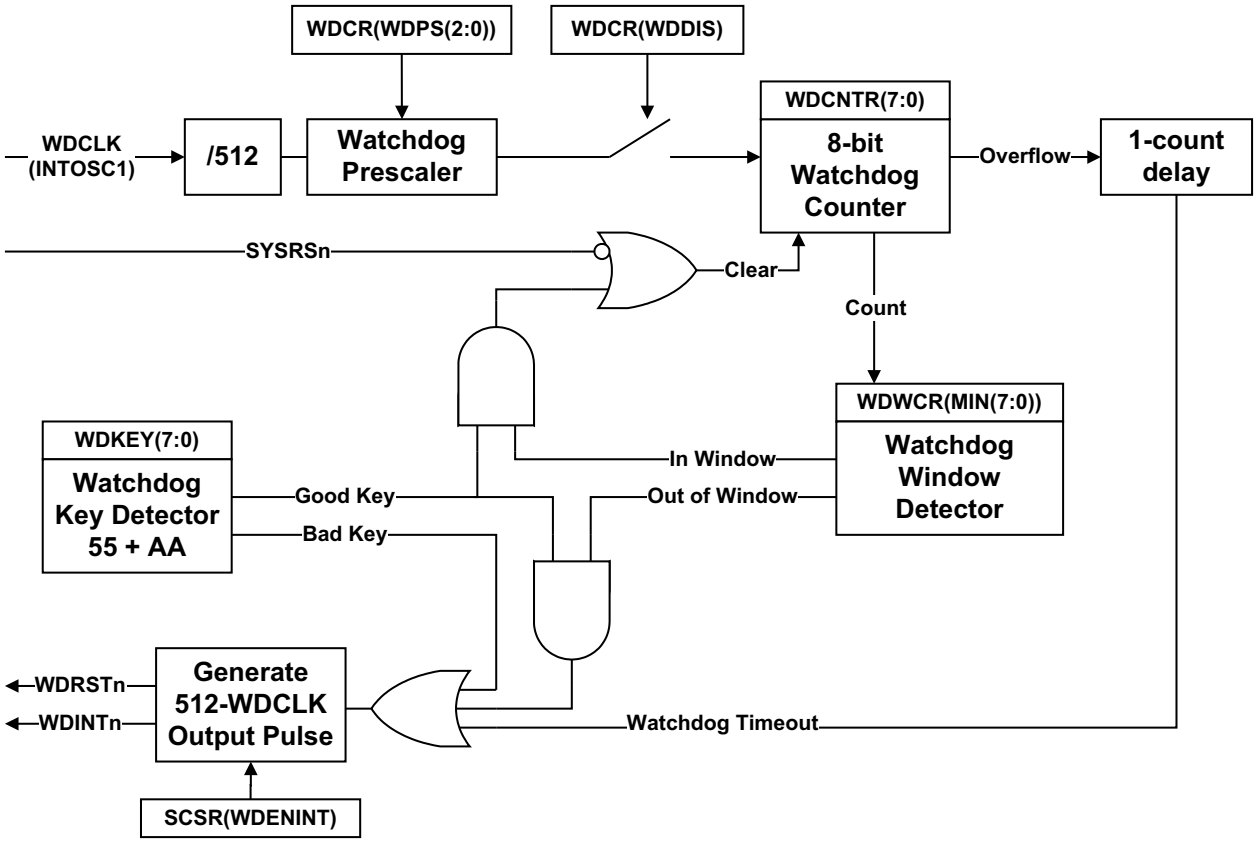



**Figure 3-13. CPU Watchdog Timer Module**



Copyright © 2024 Texas Instruments Incorporated

### 3.9.1 Servicing the Watchdog Timer

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

**Table 3-9. 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.<br>No action, WDCNTR no longer enabled to be reset by next 0xAA. |
| 12 | 0xAA | No action due to previous invalid value. |
| 13 | 0x55 | WDCNTR is enabled to be reset if next value is 0xAA. |
| 14 | 0xAA | WDCNTR is reset. |

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

If the watchdog is configured to reset the device, then a WDCR overflow or writing the incorrect value to the WDCR[WDCHK] bits 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 must clear WDRSn to allow subsequent watchdog resets to be detected. Watchdog resets are not prevented when the flag is set.

### 3.9.2 Minimum Window Check

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

To set the window minimum, write the desired minimum watchdog count to the WDWCR register. This value 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.



Copyright © 2024 Texas Instruments Incorporated

### 3.9.3 Watchdog Reset or Watchdog Interrupt Mode

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

- **Reset mode:**

  If the watchdog is configured to reset the device, then the $\overline{\text{WDRST}}$ signal pulls the device reset ($\overline{\text{XRS}}$) pin low for 512 INTOSC1 cycles when the watchdog counter reaches the maximum value.

  **Note:** After a CPU1 watchdog reset, the boot ROMs clears all of the system and message RAMs on both CPUs. After a CPU2 watchdog reset, CPU2's boot ROM clears all of the CPU2 system and message RAMs.
- **Interrupt mode:**

  When the watchdog counter expires, the watchdog asserts an interrupt by driving the $\overline{\text{WDINT}}$ signal low for 512 INTOSC1 cycles. The falling edge of $\overline{\text{WDINT}}$ triggers a WAKEINT interrupt in the PIE, if the interrupt is enabled. Because the PIE is edge-triggered, re-enabling the WAKEINT while $\overline{\text{WDINT}}$ is active does not produce a duplicate interrupt.

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

### 3.9.4 Watchdog Operation in Low-Power Modes

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

In STANDBY mode, all of the clocks to the peripherals are turned off within the CPU subsystem. The only peripheral that remains functional is the watchdog since the watchdog module runs off the oscillator clock (OSCCLK). The $\overline{\text{WDINT}}$ signal is applied to the Low Power Modes (LPM) block so that the signal can be used to wake the CPU from STANDBY low-power mode. This feature is enabled by setting LPMCR.WDINTE = 1. See Section 3.10 for details.

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

In HALT mode, the internal oscillators and CPU1 watchdog are kept active if the user sets CLKSRCCTL1.WDHALTI = 1. A watchdog reset can wake the system from HALT mode, but a watchdog interrupt cannot.

### 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. |



Copyright © 2024 Texas Instruments Incorporated

## 3.10 Low-Power Modes

This device has three clock-gating, low-power modes, and a special power-gating mode. All low-power modes are entered by setting the LPMCR register and executing the IDLE instruction. More information about the IDLE instruction can be found in the *TMS320C28x CPU and Instruction Set Reference Guide*. Note that each CPU has an LPMCR register.

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

The application can verify the following before entering STANDBY or HALT mode:

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

### *3.10.1 IDLE*

IDLE is a standard feature of the C28x CPU. In this mode, the CPU clock is gated while all peripheral clocks are left running. IDLE can thus be used to conserve power while a CPU is waiting for peripheral events. When one CPU is in IDLE, there is no effect on the other CPU subsystem.

Any enabled interrupt wakes up the CPU from IDLE mode.

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

### *3.10.2 STANDBY*

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

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

**To enter STANDBY mode:**

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

**To wake up from Standby mode:**

1. Configure the desired GPIO to trigger the wakeup.
2. Drive the selected GPIO signal low; 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 WAKEINT interrupt can also triggered by IPCINT1 sent from the other CPU and a watchdog interrupt.

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



Copyright © 2024 Texas Instruments Incorporated

If CPU2 is in STANDBY mode, writing a 1 to the RESET bit of the CPU2RESCTL register has no effect. CPU2 can be reset by any chip-level reset (POR, XRSn, CPU1.WDRSn, or CPU1.NMIWDRSn) or HIBRESETn. Alternately, CPU2 can be woken up by any configured wake-up event.

If CPU2 is in STANDBY mode and the debugger is connected, executing a debug reset on CPU2 has no effect. To wake up the CPU2 with the debugger, Click Run, Single Step, or Step over in the Debug toolbar. CCS IDE prompts the user requesting to bring the CPU out of the low-power mode. Click Yes. This wakes up CPU2 from STANDBY and continues execution.

### *3.10.3 HALT*

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

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

**To enter HALT mode:**

1. Disable all interrupts with the exception of the WAKEINT interrupt on both CPUs. The other interrupts can be reenabled after the device is brought out of HALT mode.
2. Put CPU2 into IDLE mode. (Using STANDBY causes a duplicate WAKEINT on CPU2). CPU1 must verify this by checking the LPMSTAT register.
3. Set LPMCR.LPM to 0x2. Set GPIOLPMSEL0 and GPIOLPMSEL1 to connect the chosen GPIOs to the LPM module.
4. Set CLKSRCCTL1.WDHALTI to 1 to keep the CPU1 watchdog active and INTOSC1 and INTOSC2 powered up in HALT.
5. Set CLKSRCCTL1.WDHALTI to 0 to disable the CPU1 watchdog and power down INTOSC1 and INTOSC2 in HALT.
6. Execute the IDLE instruction on CPU1 to enter HALT.

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

**Note**

Before entering HALT mode, if the system PLL is locked (SYSPLL.LOCKS = 1), the system 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 5 µs. This activates the CPU1.WAKEINT and CPU2.WAKEINT PIE interrupts.
2. Drive the wake-up GPIO high again to initiate the powering up of the SYSPLL and AUXPLL
3. Wait 16 µs 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.



Copyright © 2024 Texas Instruments Incorporated

**Table 3-10. LPM Entry and Exit Criteria**

| LPM Mode | Entry Sequence | Exit Criteria |
|---|---|---|
| CPU1 IDLE | 1. CPU1 configures CPU1.LPMCR.LPM to 00b<br>2. CPU1 Executes IDLE instruction | 1. POR/BOR<br>2. IORSn<br>3. Any enabled CPU1 interrupt<br>4. CPU1 NMI |
| CPU1 STANDBY | 1. CPU1 configures CPU1.LPMCR.LPM bits to 01b<br>2. CPU1 programs CPU1.LPMCR.QUAL bits to configure appropriate OSCCLK qualification for GPIO wakeup pin<br>3. CPU1 programs CPU1.GPIOLPMSEL0/1 register to select GPIO0 to GPIO63 pin for wake<br>4. CPU1 Executes IDLE instruction | 1. POR/BOR<br>2. IORSn<br>3. CPU1.WDINTn<br>4. NMI fired to CPU2<br>5. GPIO0.async to GPIO63.async selected using CPU1.GPIOLPMSEL0/1 (after OSCCLK qualification)<br>6. CPU2toCPU1IPCINT1<br>7. CPU1.POWERABORT |
| CPU2 IDLE | 1. CPU2 configures CPU2.LPMCR.LPM to 00b<br>2. CPU2 Executes IDLE instruction | 1. POR/BOR<br>2. IORSn<br>3. Any enabled CPU2 interrupt<br>4. CPU2 NMI |
| CPU2 STANDBY | 1. CPU2 configures CPU2.LPMCR.LPM bits to 01b<br>2. CPU2 programs CPU2.LPMCR.QUAL bits to configure appropriate OSCCLK qualification for GPIO wakeup pin<br>3. CPU2 programs CPU2.GPIOLPMSEL0/1 registers to select GPIO0 to GPIO63 pin for wake<br>4. CPU2 Executes IDLE instruction | 1. POR/BOR<br>2. IORSn<br>3. CPU2.WDINTn<br>4. NMI Fired to CPU1<br>5. GPIO0.async to GPIO63.async selected using CPU2.GPIOLPMSEL0/1 (after OSCCLK qualification)<br>6. CPU1toCPU2IPCINT1<br>7. CPU2.POWERABORT |
| HALT | 1. CPU1 configures CPU1.LPMCR.LPM bits to 10b<br>2. CPU1 programs CPU1.GPIOLPMSEL0/1 registers to select GPIO0 to GPIO63 pin for wake<br>3. CPU1 software confirms whether CPU2 has gone to IDLE or not using LPMSTAT register<br>4. CPU1 Executes IDLE instruction | 1. POR/BOR<br>2. IORSn<br>3. CPU1.WDRSn<br>4. GPIO0.async to GPIO63.async selected using CPU1.GPIOLPMSEL0/1 (kept low until XTAL, INTOSCx, and PLL powers up)<br>5. CPU1.POWERABORT<br>6. CPU2.POWERABORT |



Copyright © 2024 Texas Instruments Incorporated

### 3.10.4 Hibernate (HIB)

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

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

**To enter HIB mode:**

1. Save any necessary state to the M0 and M1 memories of both CPUs.
2. Put all I/Os in the desired state for isolation and deactivate any analog modules in use.
3. Write the address of the I/O restore function for each CPU to the IORESTOREADDR register.
4. Put CPU2 in reset, IDLE, or STANDBY.
5. Bypass the PLL by setting PLLCLKEN to 0.
6. Set CPU1's LPMCR.LPM to 0x3 and execute the IDLE instruction.

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

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

**To wake the device from HIB mode:**

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

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

**Note**

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

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



Copyright © 2024 Texas Instruments Incorporated

## 3.11 Memory Controller Module

**Note**

All RAMs on these devices are SRAMs.

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

- dedicated to each CPU (M0, M1, and Dx RAMs),
- shared between the CPU and its own CLA (LSx RAM),
- shared between the CPU and DMA of both subsystems (GSx RAM), and
- used to send and receive messages between processors (MSGRAM).

All these RAMs are highly configurable to achieve control for write access and fetch access from different masters. There are also RAMs - called IPC MSGRAMs - that are used for interprocessor communication. All dedicated RAMs are enabled with the ECC feature (both data and address) and shared RAMs, as well as IPC MSGRAMs, are enabled with the PARITY (both data and address) feature. Some of the dedicated memories are secure memory as well. Refer to Section 3.13 for more details. Each RAM has its own controller that takes care of the access protection/security related checks and ECC/Parity features for that RAM. Figure 3-14 shows the configuration of these RAMs.

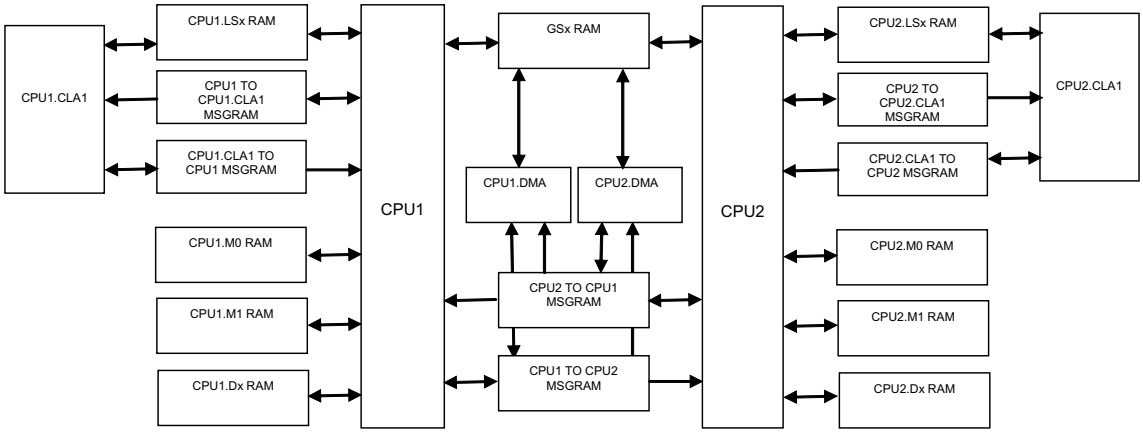



**Figure 3-14. Memory Architecture**

### *3.11.1 Functional Description*

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

#### 3.11.1.1 Dedicated RAM (Dx RAM)

Each CPU subsystem has four dedicated RAM blocks: M0, M1, D0, and D1. M0/M1 memories are small blocks of memory which are tightly coupled with the CPU. Only the CPU has access to these memories. No other masters (including DMA) have any access to these memories.

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

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

RAM blocks that are dedicated to each subsystem and are accessible to the CPU and CLA only, are called local shared RAMs (LSx RAMs). All such memories are secure memory and have the parity feature. By default, these memories are dedicated to the CPU only, and the user can choose to share these memories with the CLA by appropriately configuring the MSEL_LSx bit field in the LSxMSEL register. Further, when these memories are shared between the CPU and CLA, the user can choose to use these memories as CLA program memory by



Copyright © 2024 Texas Instruments Incorporated

configuring the CLAPGM_LSx bit field in the LSxCLAPGM registers. CPU access to all memory blocks, which are programmed as CLA program memory, are blocked.

All these RAMs have the access protection (CPU write/CPU fetch) feature. Each type of access protection for each RAM block can be enabled or disabled by configuring the specific bit in the local shared RAM access protection registers, mapped to each CPU subsystem. Table 3-11 shows the LSx RAM features.

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

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

#### 3.11.1.3 Global Shared RAM (GSx RAM)

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

Table 3-12 shows the features of the GSx RAM.

**Table 3-12. Global Shared RAM**

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

**Note**

Emulation/Debugger access is allowed from both CPUs, irrespective of the GSxMSEL setting.

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

Master select and access protection configuration for each GSx RAM block can be individually locked by the user to prevent further update to these bit fields. The user can also choose to permanently lock the configuration to individual bit fields by setting the specific bit fields in the GSxCOMMIT register (refer to the register description for more details). Once configuration is committed for a particular GSx RAM block, the configuration cannot be changed further until CPUx. $\overline{\text{SYSRS}}$ is issued. Only the CPU1 software can change the master select configuration by writing into the GSxMSEL register, mapped on the CPU1. The GSxMSEL register, which is mapped to the CPU2 subsystem, is a status register that can only be used by CPU2 software to know the master ownership for each GSx RAM block.

#### 3.11.1.4 CPU Message RAM (CPU MSG RAM)

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

This RAM has parity.



Copyright © 2024 Texas Instruments Incorporated

#### 3.11.1.5 CLA Message RAM (CLA MSGRAM)

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

This RAM has parity.

#### 3.11.1.6 Access Arbitration

For a shared RAM, multiple accesses can happen at a given time. The maximum number of accesses to any shared RAM at any given time depends on the type of shared RAM. On this device, a combination of a fixed and round robin scheme is followed to arbitrate multiple access at any given time. Accesses from the same masters are arbitrated in a fixed priority manner, but the accesses from different masters are arbitrated using the round robin scheme.

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

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

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

1. Data Write
2. Data Read/Program Fetch

Figure 3-15 represents the arbitration scheme on global shared memories:

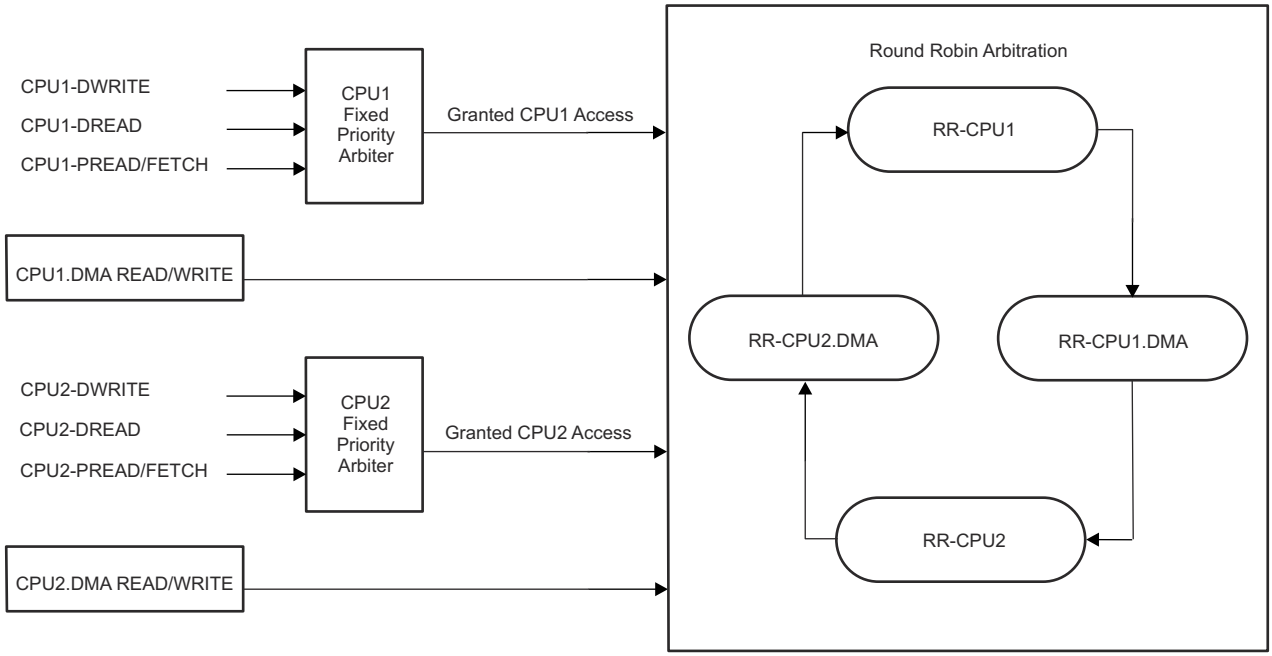



**Figure 3-15. Arbitration Scheme on Global Shared Memories**

Figure 3-16 represents the arbitration scheme on local shared memories.



Copyright © 2024 Texas Instruments Incorporated

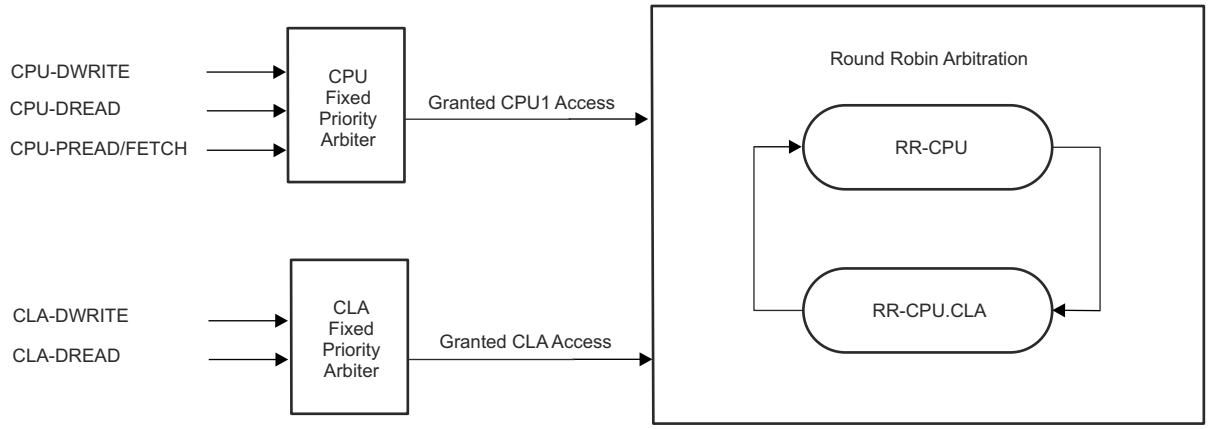



**Figure 3-16. Arbitration Scheme on Local Shared Memories**

#### 3.11.1.7 Access Protection

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

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

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

##### *3.11.1.7.1 CPU Fetch Protection*

A CPU has execution permission from a memory, only if that memory is dedicated to that CPU or its respective subsystem is master for that memory (in case of GSx memory). When fetch accesses are allowed based on the mastership, it can be further protected by setting the FETCHPROTx bit of the specific register to '1.' If fetch access is done by the CPU to a memory where CPU fetch protection is enabled, a fetch protection violation occurs.

There are two types of fetch protection violations:

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

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

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

##### *3.11.1.7.2 CPU Write Protection*

A CPU has write permission to a memory only if that memory is dedicated to that CPU, or if the respective subsystem is the master for that memory (in case of GSx memory). When write accesses are allowed based on the mastership, write accesses can be further protected by setting the CPUWRPROTx bit of the specific register to 1. If write access is done by a CPU to memory where the write is protected, a write protection violation occurs.

There are two types of CPU write protection violations:

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



Copyright © 2024 Texas Instruments Incorporated

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

#### *3.11.1.7.3 CPU Read Protection*

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

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

#### *3.11.1.7.4 CLA Fetch Protection*

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

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

#### *3.11.1.7.5 CLA Write Protection*

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

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

#### *3.11.1.7.6 CLA Read Protection*

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

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

#### *3.11.1.7.7 DMA Write Protection*

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

There are two types of DMA write protection violations:

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



Copyright © 2024 Texas Instruments Incorporated

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

If a write protection violation occurs on CPU1, write is ignored and a DMAERR interrupt gets generated, whereas in the case of CPU2, a write is ignored and an access violation interrupt is generated if enabled in the interrupt enable register. A flag gets set in the DMA access violation flag register, and the memory address where the violation happened gets latched in the DMA fetch access violation address register. These are dedicated registers for each subsystem.

**Note 1:** All access protections are ignored during debug accesses. Write access to a protected memory goes through when the write is done using the debugger, irrespective of the write protection configuration for that memory.

**Note 2:** Access protection is not implemented for M0 and M1 memories.

**Note 3:** In the case of local shared RAM, if memory is shared between the CPU and the CLA, the CPU only has access if the memory is configured as data RAM for the CLA. If the memory is programmed as program RAM, all the access from the CPU (including read) and data access from the CLA is blocked, and the violation is considered as a non-master access violation. If the memory is configured as dedicated to the CPU, all access from the CLA is blocked and the violation is considered a non-master access violation.

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

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

In this device, all dedicated RAMs support error correction code (ECC) protection and the shared RAMs have parity protection. The ECC scheme used is Single Error Correction Double Error Detection (SECDED). 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 then calculated. ECC/Parity is written into the memory along with the data. ECC/Parity is computed for 16-bit data; hence, for each 32-bit data, there are three 7-bit ECC codes (or 3-bit parity), two of which are for data and a third one for the address.

##### *3.11.1.8.1 Error Detection and Correction*

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

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

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



Copyright © 2024 Texas Instruments Incorporated

**Note**

ECC/Parity for address is calculated for address offset only (based on RAM block size) of corresponding 32-bit aligned address. For example, in case of LSx RAM that are 4-KB RAM block, only 11 LSBs of 32-bit aligned address are used. So if address is 0x8F8F, address ECC (or Parity)is calculated for address 0x78E (11-bit offset of 32-bit aligned address). Similarly for 8-KB RAM block, 12-bit address offset is used.

#### *3.11.1.8.2 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 into the correctable error threshold register, an interrupt is generated to the respective CPU, that is, if the interrupt is enabled in the correctable interrupt enable register. The user needs to configure the correctable error threshold register based on the system requirements. Also, the address for which the error occurred, gets latched into the master-specific status register and a flag gets set. Each of these registers are dedicated for each CPU subsystem.

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

Table 3-13 summarizes different error situations that can arise. These need to be handled appropriately in the software, using the status and interrupt indications provided.

**Table 3-13. Error Handling in Different Scenarios**

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

**Note**

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

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



Copyright © 2024 Texas Instruments Incorporated

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

Since error detection and correction logic is part of safety critical logic, safety applications 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.

Table 3-14 shows the bit mapping for the ECC bits when the bits are read in RAMTEST mode using their respective addresses. Table 3-15 shows the bit mapping for the Parity bits when the bits are read in RAMTEST mode using their respective addresses.

**Table 3-14. 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 3-15. 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 |

#### 3.11.1.10 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, the software must 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.

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

**Note**

None of the masters must 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.



Copyright © 2024 Texas Instruments Incorporated

## 3.12 Flash and OTP Memory

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

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

### *3.12.1 Features*

Features of Flash memory include:

- Dedicated Flash bank in the CPU1 subsystem (refer to the device data sheet for the size of Flash bank)
- Dedicated Flash bank in the CPU2 subsystem (refer to the device data sheet for the size of Flash bank)
- Dedicated Flash module controller (FMC) in the CPU1 and CPU2 subsystems for each bank
- 128 bits (bank width) can be programmed at a time along with ECC
- Multiple sectors providing the option of leaving some sectors programmed and only erasing specific sectors
- User-programmable OTP memory locations for configuring security, OTP boot-mode and boot-mode select pins (if the user is unable to use the factory-default boot-mode select pins)
- Single-Flash pump shared by the CPU1 and CPU2 subsystems
- Hardware Flash pump semaphore to control ownership of the pump between the two FMCs.
- Enhanced performance using the code-prefetch mechanism and data cache in CPU1-FMC and CPU2-FMC
- Configurable wait states to give the best performance for a given execution speed
- Safety Features
  - SECDED-single error correction and double error detection is supported in both FMCs
  - Address bits are included in ECC
  - Test mode to check the health of ECC logic
- Supports low-power modes for Flash bank and pump for power savings
- Built-in power mode control logic
- Integrated Flash program/erase state machine (FSM) in both FMCs
  - Simple Flash API algorithms
  - Fast erase and program times (refer to the device data sheet for details)
- Code Security Module (CSM) to prevent access to the Flash by unauthorized persons (refer to Section 3.13 for details)

### *3.12.2 Flash Tools*

Texas Instruments provides the following tools for Flash:

- Code Composer Studio (CCS) IDE - the development environment with integrated Flash plugin
- F021 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 IDE is required.
- CCS On-Chip Flash Plugin and UniFlash tools developed for these devices support AutoEccGeneration (see *TMS320F2837xD Flash API Version 1.54 Reference Guide*). But the tools do not support the program of ECC generated by the linker -ecc options.

  Users must check and install available updates for CCS On-Chip Flash Plugin and UniFlash tools.



Copyright © 2024 Texas Instruments Incorporated

### 3.12.3 Default Flash Configuration

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

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

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

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

**Note**

Before initializing wait-states, turn off the pre-fetch and data caching in the FRD_INTF_CTRL register.

### 3.12.4 Flash Bank, One-Time Programmable (OTP) Memory, and Flash Pump

There is a dedicated Flash bank in the CPU1 subsystem called the CPU1 Flash bank and a dedicated Flash bank in the CPU2 subsystem called the CPU2 Flash bank. Also, there is a one-time programmable (OTP) memory on the CPU1 subsystem called USER OTP, which the user can program only once and cannot erase. Flash and OTP memory are uniformly mapped in both program and data memory space.

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

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

Figure 3-23 depicts the user-programmable OTP memory locations in CPU1 USER-OTP and CPU2 USER-OTP. For more information on the functionality of these fields, refer to Section 3.13 and Chapter 4.



Copyright © 2024 Texas Instruments Incorporated

### 3.12.5 Flash Module Controller (FMC)

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

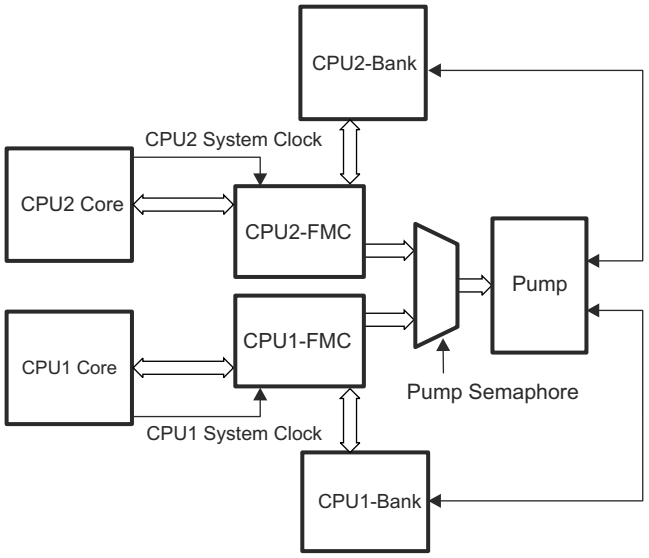



**Figure 3-17. FMC Interface with Core, Bank, and Pump**

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

There is a state machine in both CPU1-FMC and CPU2-FMC that generates the erase and program sequences in hardware. This simplifies the Flash API software that configures control registers in the FMC to perform Flash erase and program operations (see *TMS320F2837xD Flash API Version 1.54 Reference Guide*, for details on Flash API).

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

The Flash bank and pump consume a significant amount of power when active. The Flash module provides a mechanism to power-down Flash banks and pump. Special timers automatically sequence the power-up of the CPU1 Flash bank and CPU2 Flash bank independently of each other. The shared charge pump module has an independent power-up timer as well.

The Flash bank and OTP memory operate in three power modes: Sleep (lowest power), Standby, and Active (highest power)

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



Copyright © 2024 Texas Instruments Incorporated

The charge pump operates in two power modes:

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

Any access to any Flash bank/OTP memory causes the charge pump to go into active mode, if the charge pump is in sleep mode. An erase or program command causes the charge pump and bank to become active. If any bank is in active or in standby mode, the charge pump is in active mode, independent of the pump power mode control configuration (PMPPWR bit field in the FPAC1 register).

To power down the Flash pump, both the CPU1 and CPU2 must each power down the Flash pump without any Flash accesses in between. The Flash pump does not enter low-power mode if the following sequence is not followed.

1. When the system is ready to power down the Flash completely, synchronize CPU1 and CPU2. CPU2 enters the Flash power-down phase (steps 2, 3, 4, and 5) while CPU1 is waiting for CPU2 to complete.
2. Acquire the Pump Semaphore with CPU2.
3. Assign a value of 0x14 to CPU2 VREADST (refer to the FBAC register) to make sure the requisite delay is needed for the Flash pump/bank to come out of low-power mode later: FBAC.VREADST = 0x14
4. Change the CPU2 Flash Bank Fall Back power mode to Sleep: FBFALLBACK.BNKPWR = 0.
5. Change the CPU2 Flash Charge Pump Fall Back power mode to Sleep: FPAC1.PMPPWR = 0. CPU2 must notify CPU1 that CPU2 has completed the above sequence. CPU2 must wait until CPU1 completes steps 6, 7, 8, 9, and 10.
6. Acquire the Pump Semaphore with CPU1.
7. Assign a value of 0x14 to CPU1 VREADST (refer to FBAC register) to make sure the requisite delay is needed for the Flash pump/bank to come out of low-power mode later: FBAC.VREADST = 0x14
8. Change the CPU1 Flash Bank Fall Back power mode to Sleep: FBFALLBACK.BNKPWR = 0.
9. Change the CPU1 Flash Charge Pump Fall Back power mode to Sleep: FPAC1.PMPPWR = 0.
10. Release the Pump Semaphore from the CPU1. CPU1 must notify CPU2 that CPU1 has completed the power-down sequence so that both subsystems can continue.

The previous procedure must be executed from RAM and not from Flash. Note that exclusive control of the Flash pump must be gained by a CPU (using Flash pump semaphore PUMPREQUEST) before configuring the PMPPWR bit field of the FPAC1 register as shown in the previous sequence. As the charge pump is shared between CPU1-FMC and CPU2-FMC, the effective PMPPWR value used when powering down the pump is of the FMC (out of CPU1-FMC and CPU2-FMC) that owns the pump. The application software can check the current power mode of the Flash bank by reading the FBPRDY register. The PUMPRDY bit in the FBPRDY register in CPU1-FMC and CPU2-FMC together reflect the power mode of the charge pump. A value of 0 in the PUMPRDY bit in both CPU1-FMC and CPU2-FMC indicates that the charge pump is in sleep mode. A value of 1 in the PUMPRDY bit in either CPU1-FMC or CPU2-FMC or in both CPU1-FMC and CPU2-FMC indicates that the charge pump is in active mode. Refer to the register descriptions, Section 3.17, for detailed information.

While the pump is in sleep state, a charge pump sleep down counter holds a user configurable value (PSLEEP bit field in the FPAC1 register) and when the charge pump exits sleep power mode, the down counter delays from 0 to PSLEEP prescaled SYSCLK clock cycles (prescaled clock is SYSCLK/2) before putting the charge pump into active power mode. Note that the configured PSLEEP value must yield at least a delay of 20 µs for the pump to go to active mode. Refer to the register descriptions, Section 3.17, for detailed information.

Following are the number of cycles for the Bank and pump to wake up from low-power modes:

- Pump sleep to active = PSLEEP * (SYSCLK/2) cycles
- Bank sleep to standby = 425 Flash clock cycles
- Bank standby to active = 90 Flash clock cycles

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



Copyright © 2024 Texas Instruments Incorporated

### 3.12.7 Flash and OTP Memory Performance

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

RWAIT defines the number of random access wait-states and is configurable using the RWAIT bit-field in the FRDCNTL register. At reset, the RWAIT bit-field defaults to a worst-case wait-state count (15), and therefore needs to be initialized for the appropriate number of wait states to improve performance, based on the CPU clock rate and the access time of the Flash. The Flash supports 0-wait accesses when the RWAIT bits are set to zero. This assumes that the CPU speed is low enough to accommodate the access time.

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

RWAIT = ceiling[(SYSCLK/FCLK)-1]

where SYSCLK is the system operating frequency

FCLK is Flash clock frequency. FCLK must be ≤ $FCLK_{max}$, allowed maximum Flash clock frequency at RWAIT=0.

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

### 3.12.8 Flash Read Interface

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

#### 3.12.8.1 FMC Flash Read Interface

##### 3.12.8.1.1 Standard Read Mode

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

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

Standard read mode is the recommended mode for lower system-frequency operation in which RWAIT can be set to zero to provide single-cycle access operation. The FMC can operate at higher frequencies using standard read mode at the expense of adding wait states. At higher system frequencies, it is recommended to enable cache and prefetch mechanisms to improve performance. Refer to the device specific data sheet to determine the maximum Flash frequency allowed in standard read mode (that is, maximum Flash clock frequency with RWAIT=0, $FCLK_{MAX}$).

##### 3.12.8.1.2 Prefetch Mode

Flash memory is typically used to store application code. During code execution, instructions are fetched from sequential memory addresses, except when a discontinuity occurs. Usually the portion of the code that resides in sequential addresses makes up the majority of the application code and is referred to as linear code. To improve the performance of linear code execution, a Flash prefetch-mechanism has been implemented in the FMC. Figure 3-18 illustrates how this mode functions.



Copyright © 2024 Texas Instruments Incorporated

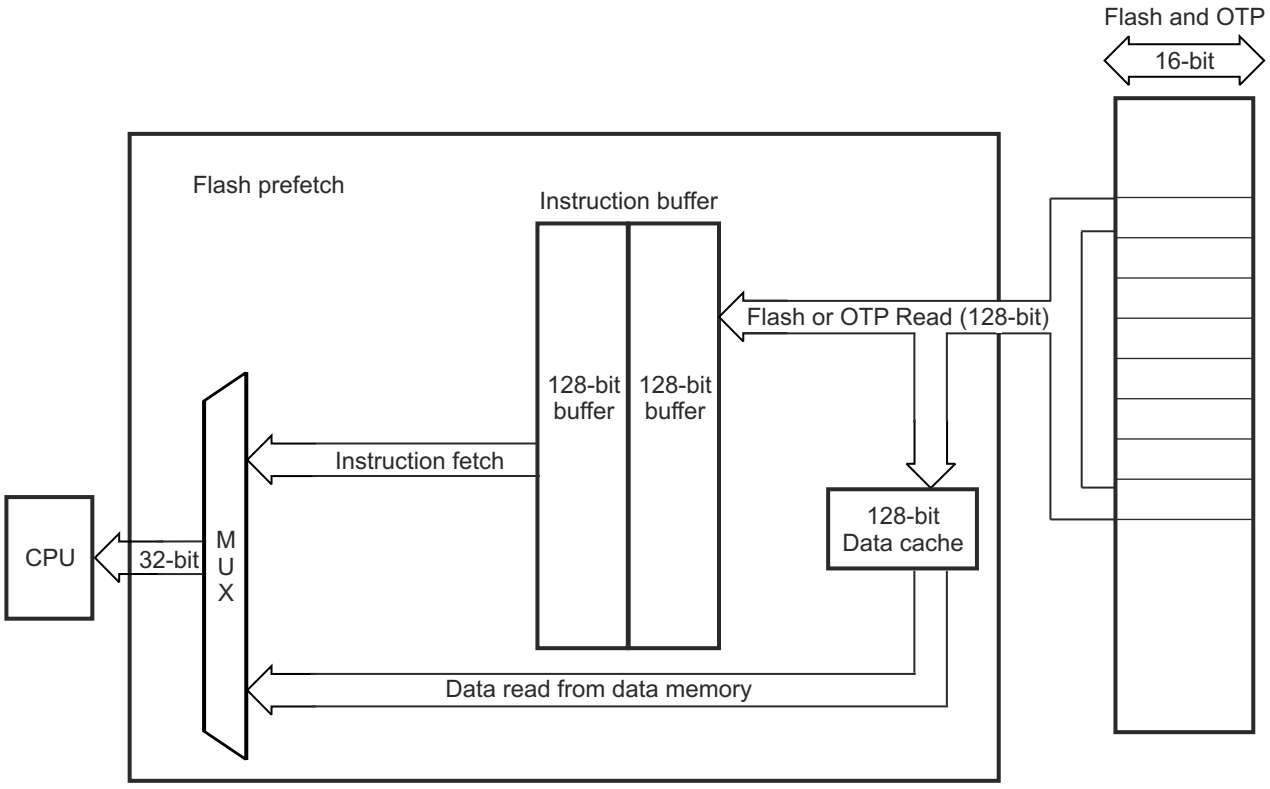



**Figure 3-18. Flash Prefetch Mode**

This prefetch mechanism does a look-ahead prefetch on linear address increments starting from the address of the last instruction fetch. The Flash prefetch mechanism is disabled by default. Setting the PREFETCH_EN bit in the FRD_INTF_CTRL register enables this prefetch mode.

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

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

**Note**

If the prefetch mechanism is enabled, then the last two rows (16 16-bit words, 256 bits) of the bank that does not have a valid address beyond the boundary must not be used, because the prefetch logic that does a look-ahead prefetch tries to fetch from outside the bank and can result in an ECC error.

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

1. If the destination address is within the Flash or OTP memory, the prefetch aborts and then resumes at the destination address.



Copyright © 2024 Texas Instruments Incorporated

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

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

#### *3.12.8.1.2.1 Data Cache*

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

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

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

**Note**

Flash contents are verified for ECC correctness before the contents enter the prefetch buffer or data cache and not inside the prefetch buffer or data cache itself.

### *3.12.9 Erase/Program Flash*

Flash memory can be programmed either by using the CCS Flash plugin or by using UniFlash. If these methods are not feasible in an application, the API can be used. The Flash memory must be programmed, erased, and verified only by using the F021 Flash API library. These functions are written, compiled and validated by Texas Instruments. The Flash module contains a Flash state machine (FSM) to perform program and erase operations. This section only provides a high level description for these operations, therefore, refer to the *TMS320F2837xD Flash API Version 1.54 Reference Guide* for more information. Note that Flash API execution is interruptible. However, there must not be any read/fetch access from the Flash bank on which an erase/program operation is in progress. Flash API must be executed from RAM.



Copyright © 2024 Texas Instruments Incorporated

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

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

#### 3.12.9.1 Erase

When the target Flash is erased, Flash reads as all 1s. This state is called 'blank.' The erase function must be executed before programming. The user must not 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. This command is implemented by the following Flash API function:

Fapi_issueAsyncCommandWithAddress();

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

Fapi_doBlankCheck();

#### 3.12.9.2 Program

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

This command is implemented by the following Flash API function:

Fapi_issueProgrammingCommand();

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

#### 3.12.9.3 Verify

After programming, the user must perform verify using API function Fapi_doVerify(). This function verifies the Flash contents against supplied data.

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.

### *3.12.10 Error Correction Code (ECC) Protection*

CPU1-FMC and CPU2-FMC contain an embedded single error correction and double error detection (SECDED) module. SECDED, when enabled, provides the capability to screen out memory faults. SECDED can detect and correct single-bit data errors and detect address errors/double-bit data errors. For every 64 bits of Flash/OTP memory data (aligned on a 64-bit memory boundary) that is programmed, eight ECC check bits have to be calculated and programmed in ECC memory space. Refer to the device data sheet for the Flash/OTP memory ECC memory-map. SECDED works with a total of eight user-calculated error correction code (ECC) check bits associated with each 64-bit wide data word and the corresponding 128-bit memory-aligned address. Users must program ECC check bits along with Flash data. TI recommends using the AutoEccGeneration option available in Plugin/API to program ECC. Users can use the F021 Flash API to calculate and program ECC data along with Flash data. Flash API uses hardware ECC logic in the device to generate the ECC data for the given Flash data. The Flash Plugin, the Flash programming tool integrated with Code Composer Studio IDE, uses Flash API to generate and program ECC data).

Figure 3-19 illustrates the ECC logic inputs and outputs.



Copyright © 2024 Texas Instruments Incorporated

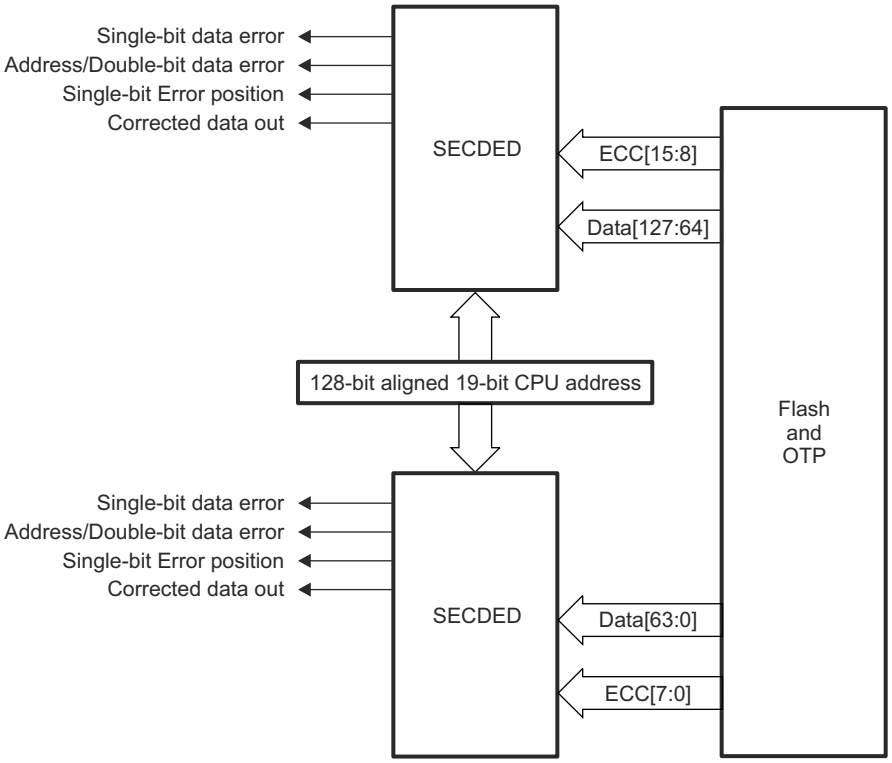



**Figure 3-19. ECC Logic Inputs and Outputs**

During an instruction fetch or a data read operation, the 19 most-significant address bits (3 least-significant bits of address are not considered), together with the 64-bit data/8-bit ECC read-out of Flash banks/ECC memory-map area, pass through the SECDED logic and the eight checkbits are produced in FMC. These eight calculated ECC check bits are then XORed with the stored check bits (user programmed check bits) associated with the address and the read data. The 8-bit output is decoded inside the SECDED module to determine one of three conditions:

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

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

**Note**

TI recommends programming ECC while programming Flash to avoid any error. Since ECC is calculated for an entire 64-bit data, a non 64-bit read such as a byte read or a half-word read still forces the entire 64-bit data to be read and calculated, but only the byte or half-word is actually used by the CPU.

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



Copyright © 2024 Texas Instruments Incorporated

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

#### 3.12.10.1 Single-Bit Data Error

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

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

- Address where the error occurred – if the single-bit error occurs in the lower 64-bits of a 128-bit memory-aligned data, the lower 64-bit memory-aligned address is captured in the SINGLE_ERR_ADDR_LOW register. If the single-bit error occurs in the upper 64-bits of a 128-bit memory-aligned data, the upper 64-bit memory-aligned address 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 data.
- Bit position at which error occurred – the ERR_POS_L and ERR_POS_H bit fields in the ERR_POS register indicate the bit position of the error in the lower 64-bits/lower 8-bit ECC, or the upper 64-bits/upper 8-bit ECC respectively, of a 128-bit memory-aligned data.
- Whether the corrected value is 0 (FAIL_0_L, FAIL_0_H flags in ERR_STATUS register)
- Whether the corrected value is 1 (FAIL_1_L, FAIL_1_H flags in ERR_STATUS register)
- A single bit error counter that increments on every single bit error occurrence (ERR_CNT register) until a user-configurable threshold (see ERR_THRESHOLD) is met
- A flag that gets set when one or more single-bit errors occurs after ERR_CNT equals ERR_THRESHOLD (SINGLE_ERR_INT_FLG flag in the ERR_INTFLG register)

When the ERR_CNT value equals THRESHOLD+1 value and a single bit error occurs, the SINGLE_ERR_INT flag is set, and an interrupt (FLASH_CORRECTABLE_ERR on C28x PIE has to be enabled for interrupt, if needed) is fired. The SINGLE_ERR interrupt is not fired again until the SINGLE_ERR_INTFLG is cleared. If the single error interrupt flag is not cleared using the corresponding error interrupt clear bit in the ERR_INTCLR register, the error interrupt does not come again, as this is an edge-based interrupt.

When multiple single-bit errors get caught by ECC logic, Flash ECC registers hold the information related to the latest ECC error. When multiple single-bit errors get caught, both FAIL_0_L and FAIL_1_L (and/or FAIL_0_H and FAIL_1_H) might get 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 memory causes the single-bit error flag to get set when there is a single-bit error in both or in either one of the lower 64 and upper 64 bits (or corresponding ECC check bits) of that 128-bit data.

#### 3.12.10.2 Uncorrectable Error

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

- Address where the error occurred – if the uncorrectable error occurs in the lower 64-bits of a 128-bit memory-aligned data, the lower 64-bit memory-aligned address 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 data, 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 data.
- A flag is set indicating that an uncorrectable error interrupt is fired (UNC_ERR_INTFLG in ERR_INTFLG register)



Copyright © 2024 Texas Instruments Incorporated

When an uncorrectable error occurs, the UNC_ERR_INTFLG bit is set and an uncorrectable error interrupt is fired. This uncorrectable error interrupt generates an NMI, if enabled. If an uncorrectable error interrupt flag is not cleared using the corresponding error interrupt clear bit in the ERR_INTCLR register, an error interrupt does not come again, as this is an edge based interrupt.

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

#### 3.12.10.3 SECDED Logic Correctness Check

Since error detection and correction logic are part of safety-critical logic, safety applications need to make sure that the SECDED logic is always working properly. For these safety concerns, make sure the correctness of the SECDED logic, an ECC test mode is provided to test the correctness of ECC logic periodically. In this ECC test mode, data/ECC and address inputs to the ECC logic are controlled by the ECC test mode registers FDATAH_TEST, FDATAL_TEST, FECC_TEST, and FADDR_TEST, respectively. Using this test mode, users can introduce single-bit errors, double-bit errors, or address errors and check whether or not SECDED logic is catching those errors. Users can also check if SECDED logic is reporting any false errors when no errors are introduced.

This ECC test mode can be enabled by setting the ECC_TEST_EN bit in the FECC_CTRL register. When ECC test mode is enabled, the CPU cannot read the data from Flash and instead the CPU gets data from the ECC test mode registers (FDATAH_TEST/FDATAL_TEST). This is because ECC test mode registers (FDATAH_TEST, FDATAL_TEST, FECC_TEST) are multiplexed with data from the Flash. Hence, the CPU must not read/fetch from Flash when ECC test mode is enabled. For this reason, ECC test mode code must be executed from RAM and not from Flash.

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

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

1. Obtain the ECC for a given Flash address (128-bit aligned) and 64-bit data by using the Auto ECC generation option provided in Flash API .
2. Develop an application to test ECC logic using the above data. In this application
   - Write the 128-bit aligned 19-bit Flash address in FADDR_TEST
   - Write 64-bit data in FDATAH_TEST (upper 32-bits) and FDATAL_TEST (lower 32-bits) registers
   - Write the corresponding 8-bit ECC in the FECC_TEST register
   - In any of the above three steps, users can insert errors (single-bit data error or double-bit data error or address error or single-bit ECC error or double-bit ECC error) to check whether or not ECC logic is able to catch the errors
   - Select the ECC logic block (lower 64-bits or upper 64-bits) which needs to be tested using the ECC_SELECT bit in the FECC_CTRL register
   - Enable ECC test mode using the ECC_TEST_EN bit in FECC_CTRL register
   - Write a value of 1 in the DO_ECC_CALC bit in FECC_CTRL register to enable ECC test logic for a single cycle to evaluate the address, data, ECC in FADDR_TEST, FDATAx_TEST and FECC_TEST registers for ECC errors

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

- The FECC_OUTH register holds the data output bits 63:32 from the SECDED block under test
- The FECC_OUTL register holds the data output bits 31:0 from the SECDED block under test
- The FECC_STATUS register holds the status of single-bit error occurrence, uncorrectable error occurrence, and error position of single- bit error in data/check bits



Copyright © 2024 Texas Instruments Incorporated

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

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

For 16-bit read: unsigned int variable = __addr32_read_uint16(unsigned long address);

For 32-bit read: unsigned long variable = __addr32_read_uint32(unsigned long address);

### *3.12.11 Reserved Locations Within Flash and OTP Memory*

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

- The entire OTP memory has reserved user-configurable locations for security and boot process. For more details on the functionality of these fields, refer to Section 3.13 and Chapter 4.
- Refer to Chapter 4 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 the user programs a branch instruction here, that then redirects code execution to the entry point of the application.

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

During Flash configuration, no accesses to the Flash or OTP memory 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 this procedure for any code that modifies the Flash control registers.

1. Start executing application code from RAM/Flash/OTP memory.
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 memory. The function must reside in RAM.
3. Execute the Flash configuration code (must be located in RAM) that writes to Flash control registers like FRDCNTL, FRD_INTF_CTRL, and so on.
4. At the end of the Flash configuration code execution, wait eight cycles to let the write instructions propagate through the CPU pipeline. This must be done before the return-from-function call is made.
5. Return to the calling function that resides in RAM or Flash/OTP memory and continue execution.

### *3.12.13 Simple Procedure to Modify an Application from RAM Configuration to Flash Configuration*

All of the C2000Ware example projects are provided with both RAM and Flash build configurations. To change the build configuration from RAM to Flash, import the project in to the CCS IDE and right click on the project and select 'Build Configurations' -> 'Set Active' -> 'Flash'. By selecting this, notice that:

1. _FLASH symbol is defined in the "Predefined symbols" section under Project Build settings. This is used to define and execute any Flash-build specific code.
2. Flash-based linker command file is chosen for the application instead of a RAM-based linker command file. Flash-based linker command files are provided in the C2000Ware for reference (for example: XXX_FLASH_lnk_cpu1.cmd at C2000Ware_x_xx_xx_xx\device_support\XXX\common\cmd). Flash-based linker command files have codestart mapped to a Flash entry point address.
3. All of the initialized sections are mapped to Flash memory in the Flash-based linker command file.
4. All of the functions that need to execute from RAM (for initialization or 0-wait performance purpose) are assigned to the .TI.ramfunc section in the code. For example, Flash_initModule() is assigned to .TI.ramfunc section.
5. TI.ramfunc section is mapped to a Flash address for “Load” and a RAM address for “RUN” in the Flash-based linker command file.
6. All of the sections mapped to Flash are aligned on a 128-bit boundary using the ALIGN() directive in the Flash-based linker command file.
7. memcpy() function is called in the application to copy the .TI.ramfunc content from Flash to RAM. The memcopy() is called before executing any code that is assigned to .TI.ramfunc section.



Copyright © 2024 Texas Instruments Incorporated

8. For EABI type executable: All uninitialized sections mapped to RAM are defined as NOINIT sections (using the directive “type=NOINIT”) in the linker cmd file.

### 3.12.14 Flash Pump Ownership Semaphore

Each CPU subsystem has a Flash bank, which the CPU subsystem can read, program, and erase. Both Flash banks share a single charge pump for program and erase operations. Hence, only one CPU can program or erase the Flash at any given time. A CPU can read data and execute code from the Flash even when the other CPU is programming or erasing. The Flash pump ownership semaphore allows one CPU to take control of the pump without being interrupted by the other CPU.

The pump ownership semaphore is implemented as a two-bit field in a PUMPREQUEST register with special write protections. This register requires a key field to be written at the same time as the semaphore bits. The possible semaphore states are:

| | |
|---|---|
| 00 or 11 | Either CPU can write to the semaphore. CPU1 has control of the resource by default. 00 is the reset state. |
| 01 | CPU2 has exclusive control of the resource and exclusive write access to the semaphore. |
| 10 | CPU1 has exclusive control of the resource and exclusive write access to the semaphore. |

Each CPU is only allowed to take control of the pump for itself. Direct transfer between the 01 and 10 states is not allowed. However, CPU1 can force both semaphores into the default state (00) at any time by putting CPU2 into reset. Figure 3-20 shows the allowed states and state transitions.

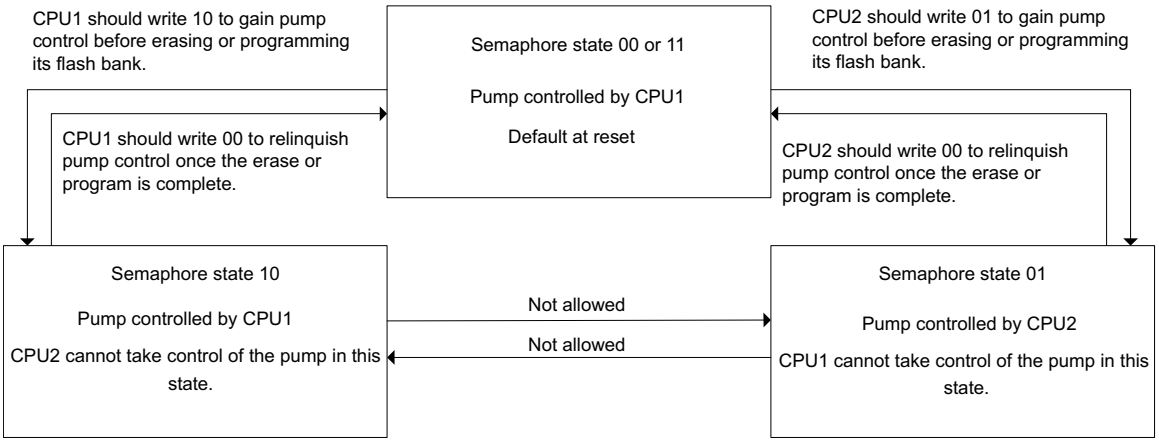



**Figure 3-20. Flash Pump Semaphore (PUMPREQUEST) States and State Transitions**



Copyright © 2024 Texas Instruments Incorporated

#### 3.12.14.1 Clock Configuration Semaphore

Both CPUs can access the PLL and peripheral clock configuration registers. The clock configuration semaphore allows one CPU to access the registers without being interrupted by the other CPU.

The clock configuration semaphore is implemented as a two-bit field in a register with special write protections. This register requires a key field to be written at the same time as the semaphore bits. The possible semaphore states are:

| | |
|---|---|
| 00 or 11 | Either CPU can write to the semaphore. CPU1 has control of the clock configuration registers by default. 00 is the reset state. |
| 01 | CPU2 has exclusive control of the clock configuration registers and exclusive write access to the semaphore. |
| 10 | CPU1 has exclusive control of the clock configuration registers and exclusive write access to the semaphore. |

Each CPU is only allowed to take control of the clock configuration registers for itself. However, CPU1 can force both semaphores into the default state (00) at any time by putting CPU2 into reset. Figure 3-21 shows the allowed states and state transitions.

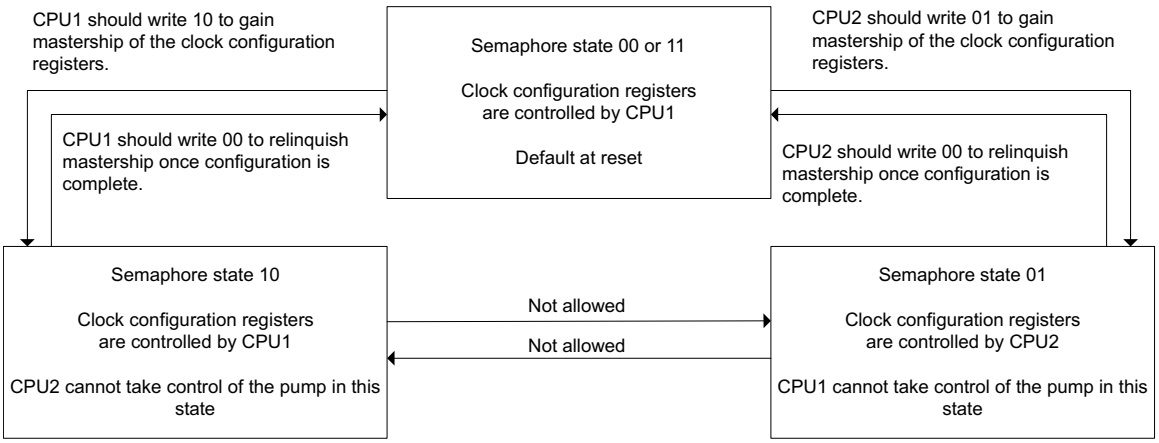



**Figure 3-21. Clock Configuration Semaphore (CLKSEM) State Transitions**



Copyright © 2024 Texas Instruments Incorporated

## 3.13 Dual Code Security Module (DCSM)

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

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

### *3.13.1 Functional Description*

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

The code security mechanism offers protection for two zones, Zone 1 (Z1) and Zone 2 (Z2). The security mechanism for both the zones is identical. Each zone has 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).
- **CLA:** The CLA is a secure resource that can be allocated to either zone by configuring the GRABRAM location in the USER OTP. CLA configuration can only be performed by code running from the zone to which it has been allocated. The CLA message RAMs also belong to the same zone.

**Table 3-16. CLA Access Filter**

| CLA Ownership | RAM Block Ownership | Fetch Access | Read Access | Write Access |
|---|---|---|---|---|
| None | None | Yes | Yes | Yes |
| None | Z1 or Z2 | No | No | No |
| Z1 | Z1 | Yes | Yes | Yes |
| Z1 | Z2 | No | No | No |
| Z1 | None | No | Yes | Yes |
| Z2 | Z1 | No | No | No |
| Z2 | Z2 | Yes | Yes | Yes |
| Z2 | None | No | Yes | Yes |

- **RAM:** All Dx and LSx RAMs can be secure RAM on this device. These RAMs can be allocated to either zone by configuring the respective GRABRAM location in the USER OTP.
- **Flash Sectors:** Flash Sectors can be secure on this device. Each Flash sector can be allocated to either zone by configuring the respective GRABSECT location in the USER OTP.
- **Secure ROM:** This device also has secure ROM which is EXEONLY-protected. This ROM contains specific function for the user, provided by TI.

Table 3-17 shows the status of a RAM block based on the configuration in GRABRAM register.



Copyright © 2024 Texas Instruments Incorporated

**Table 3-17. RAM Status**

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

The security of each zone is maintained by a 128-bit (four 32-bit words) password (CSM password). The password for each zone is stored in a dedicated OTP memory location based on a zone-specific link pointer. A zone can be unsecured by executing the password match flow (PMF), described in Section 3.13.3.3.2.

There are three types of accesses: data/program reads, JTAG access, and instruction fetches (calls, jumps, code executions, ISRs). Instruction fetches are never blocked. JTAG accesses are always blocked when a memory is secure. Data reads to a secure memory are always blocked unless the program is executing from a memory which belongs to the same zone. Data reads to unsecure memory are always allowed. Table 3-18 shows the levels of security.

**Table 3-18. Security Levels**

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

If the password locations of a zone have all 128 bits as ones, the zone is considered unsecure. Since new Flash devices have erased Flash (all ones), only a read of the password locations is required to bring any zone into unsecure mode. If the password locations of a zone have all 128 bits as zeros, the zone is secure, regardless of the contents of the CSMKEY registers. This means the zone cannot be unlocked using PMF, the password match flow described in Section 3.13.3.3.2. Therefore, the user must never use all zeros as a password. A password of all zeros prevents debug of secure code or reprogramming the Flash.

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

#### 3.13.1.1 Emulation Code Security Logic (ECSL)

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



Copyright © 2024 Texas Instruments Incorporated

When initially debugging a device with the password locations in OTP memory programmed (secured), the emulator takes some time to take control of the CPU. During this time, the CPU starts running and executes 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.

A solution to this problem is:

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

#### 3.13.1.2 CPU Secure Logic

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

#### 3.13.1.3 Execute-Only Protection

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

#### 3.13.1.4 Password Lock

The password locations for each zone can be locked by programming the zone’s PSWDLOCK field with any value other than 1111 (0xF) at the PSWDLOCK location in OTP memory. 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 memory is 1111, which means the password for all zones is unlocked.

**Note**

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

#### 3.13.1.5 JTAG Lock

The JTAG lock feature can be used to permanently disable any access to the device using the JTAG port. For example, access by the Code Composer Studio™ IDE debugger or Flash programming tools are blocked. This feature is enabled by programming the Z1OTP_JTAGLOCK location in user OTP memory with any value other than 1111 (0xF). Values of CRCLOCK, JTAGLOCK, and PSWDLOCK locations in the OTP memory are copied in the Z1_OTPSECLOCK register by the boot-ROM code.

**CAUTION**

If the JTAG lock feature is enabled, all future debug of the device through JTAG is disabled. This specifically impairs TI's ability to analyze devices returned to TI for failure analysis. If the JTAG lock feature is enabled, TI rejects any return analysis requests.



Copyright © 2024 Texas Instruments Incorporated

#### 3.13.1.6 Link Pointer and Zone Select

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

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

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

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

The location of the zone select region in OTP memory is decided based on the value of three 29-bit link pointers (Zx-LINKPOINTERx) programmed in the OTP memory of each zone of both CPU subsystems. All OTP memory locations except link pointer locations are protected with ECC. Since the link pointer locations are not protected with ECC, three link pointers are provided that need to be programmed with the same value. The final value of the link pointer is resolved in hardware when a dummy read is done to all the link pointers by comparing all the three values (bit-wise voting logic). Since in OTP memory, a 1 can be flipped by the user to 0 but a 0 cannot be flipped to a 1 (no erase operation for OTP memory), the most-significant bit position in the resolved link pointer that is 0 defines the valid base address for the zone select region. While generating the final link pointer value, if the bit patterns is not one of those listed in Figure 3-22, the final link pointer value becomes All_1 (0xFFFF_FFFF) that selects the Zone-Select-Block1 (also known as the default zone select block).



Copyright © 2024 Texas Instruments Incorporated

| Zx-LINKPOINTER | Addr Offset Of Zone-Select Block |
|---|---|
| 32'bxxx11111111111111111111111111111 | 0x20 |
| 32'bxxx11111111111111111111111111110 | 0x30 |
| 32'bxxx11111111111111111111111111100 | 0x40 |
| 32'bxxx11111111111111111111111111000 | 0x50 |
| 32'bxxx11111111111111111111111110000 | 0x60 |
| 32'bxxx11111111111111111111111100000 | 0x70 |
| 32'bxxx11111111111111111111111000000 | 0x80 |
| 32'bxxx11111111111111111111110000000 | 0x90 |
| 32'bxxx11111111111111111111100000000 | 0xa0 |
| 32'bxxx11111111111111111111000000000 | 0xb0 |
| 32'bxxx11111111111111111110000000000 | 0xc0 |
| 32'bxxx11111111111111111100000000000 | 0xd0 |
| 32'bxxx11111111111111111000000000000 | 0xe0 |
| 32'bxxx11111111111111110000000000000 | 0xf0 |
| 32'bxxx11111111111111100000000000000 | 0x100 |
| 32'bxxx11111111111111000000000000000 | 0x110 |
| 32'bxxx11111111111110000000000000000 | 0x120 |
| 32'bxxx11111111111100000000000000000 | 0x130 |
| 32'bxxx11111111111000000000000000000 | 0x140 |
| 32'bxxx11111111110000000000000000000 | 0x150 |
| 32'bxxx11111111100000000000000000000 | 0x160 |
| 32'bxxx11111111000000000000000000000 | 0x170 |
| 32'bxxx11111110000000000000000000000 | 0x180 |
| 32'bxxx11111100000000000000000000000 | 0x190 |
| 32'bxxx11111000000000000000000000000 | 0x1a0 |
| 32'bxxx11110000000000000000000000000 | 0x1b0 |
| 32'bxxx11100000000000000000000000000 | 0x1c0 |
| 32'bxxx11000000000000000000000000000 | 0x1d0 |
| 32'bxxx10000000000000000000000000000 | 0x1e0 |
| 32'bxxx00000000000000000000000000000 | 0x1f0 |

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

**Figure 3-22. Storage of Zone-Select Bits in OTP Memory**

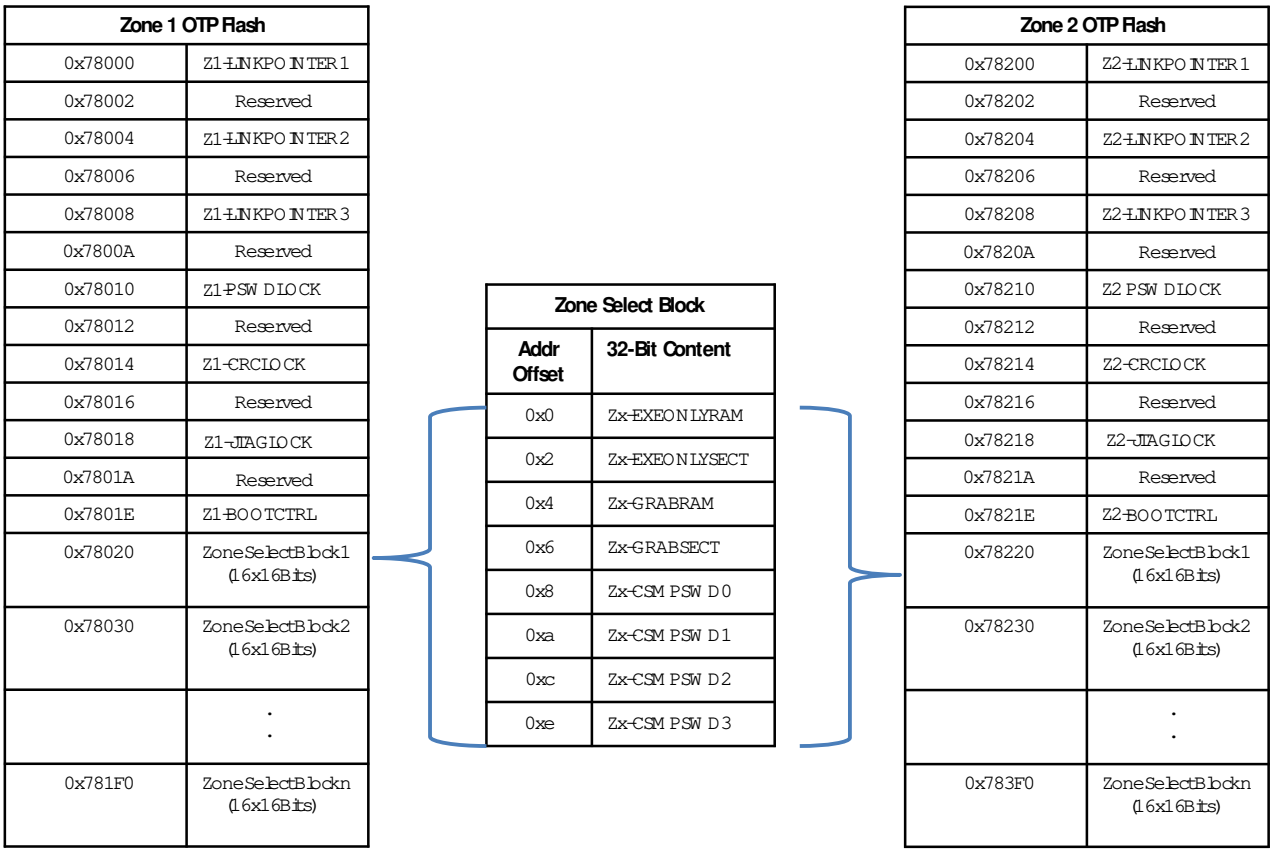
**Note**

Address locations for other security settings (PSWDLOCK/CRCLOCK) that are not part of Zone Select blocks can be programmed only once; therefore, the user must program the address locations toward the end of the development cycle.

Since linkpointer location in USER OTP does not have ECC, the user must always define a separate structure and section for linkpointers.



Copyright © 2024 Texas Instruments Incorporated



| Zone 1 OTP Flash | |
|---|---|
| 0x78000 | Z1-LINKPOINTER1 |
| 0x78002 | Reserved |
| 0x78004 | Z1-LINKPOINTER2 |
| 0x78006 | Reserved |
| 0x78008 | Z1-LINKPOINTER3 |
| 0x7800A | Reserved |
| 0x78010 | Z1-PSWDLOCK |
| 0x78012 | Reserved |
| 0x78014 | Z1-CRCLOCK |
| 0x78016 | Reserved |
| 0x78018 | Z1-JTAGLOCK |
| 0x7801A | Reserved |
| 0x7801E | Z1-BOOTCTRL |
| 0x78020 | ZoneSelectBlock1 (16x16Bits) |
| 0x78030 | ZoneSelectBlock2 (16x16Bits) |
| | . . |
| 0x781F0 | ZoneSelectBlockn (16x16Bits) |

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

| Zone 2 OTP Flash | |
|---|---|
| 0x78200 | Z2-LINKPOINTER1 |
| 0x78202 | Reserved |
| 0x78204 | Z2-LINKPOINTER2 |
| 0x78206 | Reserved |
| 0x78208 | Z2-LINKPOINTER3 |
| 0x7820A | Reserved |
| 0x78210 | Z2 PSWDLOCK |
| 0x78212 | Reserved |
| 0x78214 | Z2-CRCLOCK |
| 0x78216 | Reserved |
| 0x78218 | Z2-JTAGLOCK |
| 0x7821A | Reserved |
| 0x7821E | Z2-BOOTCTRL |
| 0x78220 | ZoneSelectBlock1 (16x16Bits) |
| 0x78230 | ZoneSelectBlock2 (16x16Bits) |
| | . . |
| 0x783F0 | ZoneSelectBlockn (16x16Bits) |

**Figure 3-23. Location of Zone-Select Block Based on Link-Pointer**

#### *3.13.1.6.1 C Code Example to get Zone Select Block Addr for Zone1*

```
unsigned long LinkPointer;
unsigned long *Zone1SelBlockPtr;
int Bitpos = 28;
int ZeroFound = 0;
// Read Z1-Linkpointer register of DCSM module.
LinkPointer = *(unsigned long *)0x5F000;
// Bits 31 30 and 29 as most-sigificant 0 are reserved LinkPointer options
LinkPointer = LinkPointer << 3;
while ((ZeroFound == 0) && (bitpos > -1))
{
    if ((LinkPointer & 0x80000000) == 0)
        {
               ZeroFound = 1;
               Zone1SelBlockPtr = (unsigned long *)(0x78000 + ((bitpos + 3)*16));
        } else
        {
               bitpos--;
               LinkPointer = LinkPointer << 1;
        }
}
if (ZeroFound == 0)
{
       //Default in case there is no zero found.
       Zone1SelBlockPtr = (unsigned long *)0x78020;
}
```



Copyright © 2024 Texas Instruments Incorporated

#### 3.13.1.7 Flash and OTP Memory Erase/Program

On this device, OTP memory, as well as normal Flash, are secure resources. Each zone has a dedicated OTP memory, whereas normal Flash sectors can be allocated to any zone based on the value programmed in the GRABSECT location in OTP memory. Each zone has CSM passwords; read and write accesses are not allowed to resources owned by Z1 from 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 cannot be updated. The OTP Flash content cannot be erased.

This device has only one Flash pump used for erase/program operation of normal Flash and OTP Flash. A semaphore mechanism is provided to avoid the conflict between Zone1 and Zone2. A zone needs to grab this semaphore to successfully complete the erase/program operation on the Flash sectors allocated to that zone. A semaphore can be grabbed by a zone by writing the appropriate value in the SEM field of the FLSEM register. For further details of this field, see the register description.

**Note**

If there is a loss of power or a reset of any nature during the Flash programming operation, there is high probability of some (or possibly all) of the 128 bits in the corresponding 128-bit aligned address getting corrupted. If this happens while programming the password locations in USER OTP, the passwords can get corrupted.

#### 3.13.1.8 Safe Copy Code

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

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

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

#### 3.13.1.9 SafeCRC

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

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

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

**Note**

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



Copyright © 2024 Texas Instruments Incorporated

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

### *3.13.2 CSM Impact on Other On-Chip Resources*

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

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

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

**Note**

Security Initialization is done by BOOTROM code on all the resets that assert SYSRSn (as part of device initialization). This is not part of the user application code.



Copyright © 2024 Texas Instruments Incorporated

### 3.13.3 Incorporating Code Security in User Applications

Code security is typically not required in the development phase of a project. However, security is needed once a robust code is developed for a zone. Before such a code is programmed in the Flash memory, a CSM password 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.15) 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.

#### 3.13.3.1 Environments That Require Security Unlocking

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

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

The unsecuring sequence is identical in all the above situations. This sequence is referred to as the password match flow (PMF) for simplicity. Figure 3-24 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.



Copyright © 2024 Texas Instruments Incorporated

#### 3.13.3.2 CSM Password Match Flow

Password match flow (PMF) is essentially a sequence of four dummy reads from password locations (PWL) followed by four writes (32-bit writes) to CSMKEY(0/1/2/3) registers. Figure 3-24 shows how PMF helps to initialize the security logic registers and disable security logic.

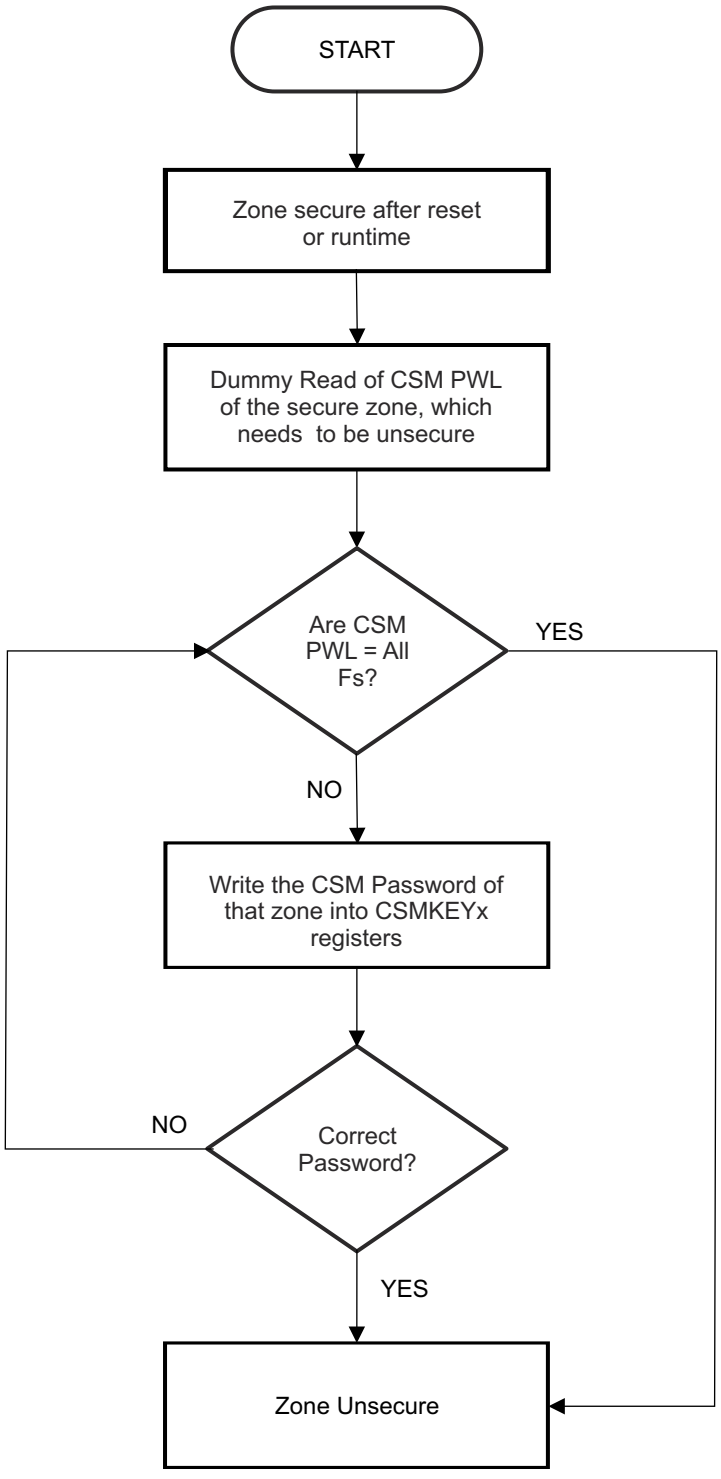



**Figure 3-24. CSM Password Match Flow (PMF)**



Copyright © 2024 Texas Instruments Incorporated

#### 3.13.3.3 Unsecuring Considerations for Zones With and Without Code Security

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

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

  A zone with code security must have a predetermined password stored in the password locations of that zone. The following are steps to unsecure any secure zone:
  1. Perform a dummy read of the password locations of that zone.
  2. Write the password into the CSMKEY registers.
  3. If the password is correct, the zone becomes unsecure; otherwise, the zone stays secure.

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

  A zone without code security must have 0x FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF (128 bits of all ones) stored in the password locations. The following are steps to use this zone:
  1. At reset, the CSM locks memory regions protected by the CSM.
  2. Perform a dummy read of the password locations.
  3. Since the password is all ones, this unlocks the zone and all the secure memories dedicated to that zone are fully accessible immediately after this operation is completed.

**Note**

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

##### *3.13.3.3.1 C Code Example to Unsecure C28x Zone1*

```
volatile long int *CSM = (volatile long int *)0x5F010;    //CSM register file
volatile long int *CSMPWL = (volatile long int *)0x78028; //CSM Password location (assuming default
Zone sel block)
                                                            volatile int tmp;
int I;
// Read the 128-bits of the CSM password locations (PWL)
//
for (I=0;I<4; I++) tmp = *CSMPWL++;
// If the password locations (CSMPWL) are all = ones (0xFFFF),
// then the zone will now be unsecure. If the password
// is not all ones (0xFFFF), then the code below is required
// to unsecure the CSM.
// Write the 128-bit password to the CSMKEY registers
// If this password matches that stored in the
// CSLPWL then the CSM will become unsecure. If it does not
// match, then the zone will remain secure.
// An example password of:
// 0x1111222233334444555566667778888 is used.
*CSM++ = 0x22221111; // Register Z1_CSMKEY0 at 0x5F010
*CSM++ = 0x44443333; // Register Z1_CSMKEY1 at 0x5F012
*CSM++ = 0x66665555; // Register Z1_CSMKEY2 at 0x5F014
*CSM++ = 0x88887777; // Register Z1_CSMKEY3 at 0x5F016
```

##### *3.13.3.3.2 C Code Example to Resecure C28x Zone1*

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

#### 3.13.3.4 Environments That Require ECSL Unlocking

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

- The user develops some main IP, and then outsources peripheral functions to a subcontractor who must be able to run the user code during debug and may halt while main IP code is running. If ECSL is not unlocked,



Copyright © 2024 Texas Instruments Incorporated

then Code Composer Studio IDE connections get disconnected, which can be inconvenient for the user. Note that unlocking ECSL doesn't enable access to secure code but only avoids disconnection of CCS (JTAG).

#### 3.13.3.5 ECSL Password Match Flow

A password match flow (PMF) is essentially a sequence of eight dummy reads from password locations (PWL) followed by two writes to KEY registers. Figure 3-25 shows how the PMF helps to initialize the security logic registers and disable security logic.

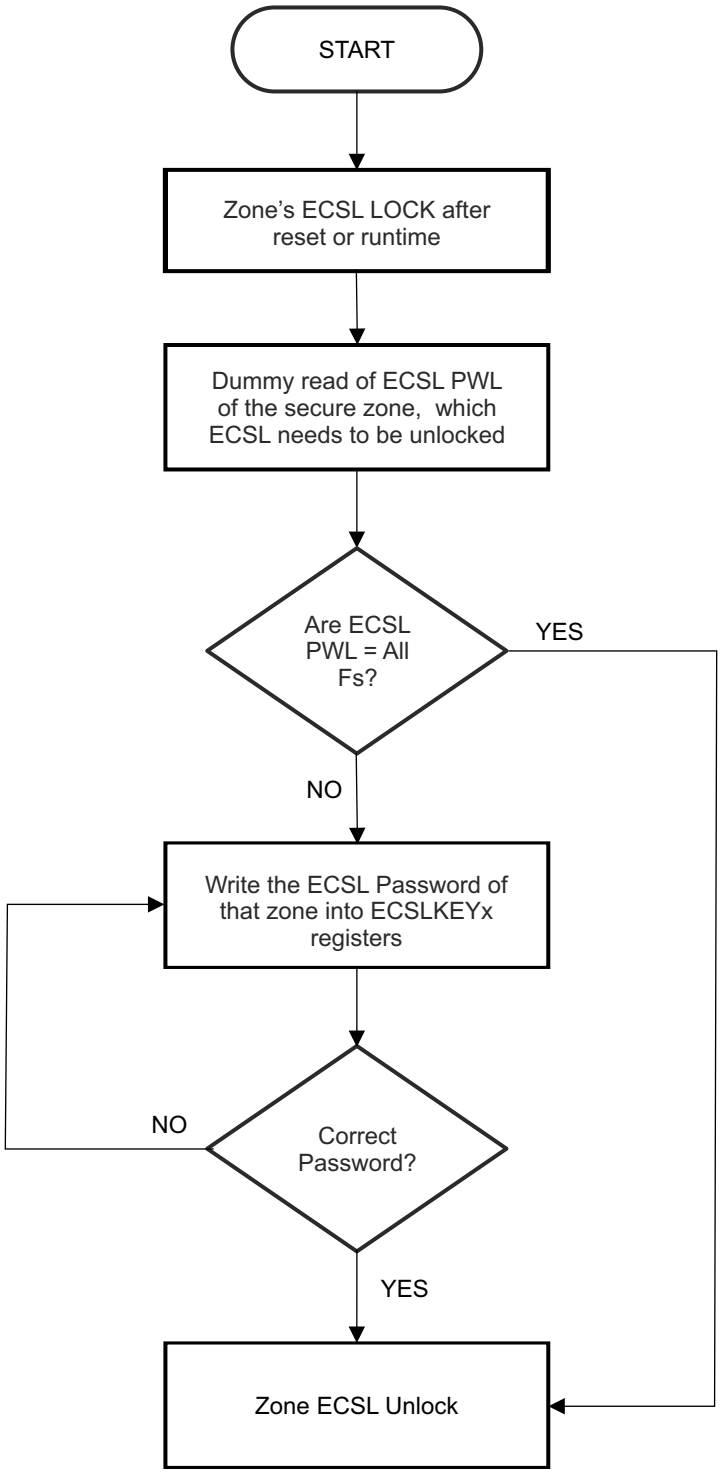



**Figure 3-25. ECSL Password Match Flow (PMF)**



Copyright © 2024 Texas Instruments Incorporated

#### 3.13.3.6 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). The following are steps to disable the ECSL for any particular zone:

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

##### *3.13.3.6.1 C Code Example to Disable ECSL for C28x-Zone1*

```
volatile long int *ECSL = (volatile int *)0x5F010;    //ECSL register file
volatile long int *ECSLPWL = (volatile int *)0x78028; //ECSL Password location (assuming default
Zone sel block)
                                                        volatile int tmp;
int I;
// Read the 64-bits of the password locations (PWL)
.
for (I=0;I<2; I++) tmp = *ECSLPWL++;
// If the ECSL password locations (ECSLPWL) are all = ones (0xFFFF),
// then the ECSL will now be disable. If the password
// is not all ones (0xFFFF), then the code below is required
// to disable the ECSL.
// Write the 64-bit password to the CSMKEYx registers
// If this password matches that stored in the
// CSMPWL then ECSL will get disable. If it does not
// match, then the zone will remain secure.
// An example password of:
// 0x1111222233334444 is used.
*ECSL++ = 0x22221111; // Register Z1_CSMKEY0 at 0x5F010
*ECSL++ = 0x44443333; // Register Z1_CSMKEY1 at 0x5F012
```

#### 3.13.3.7 Device Unique ID

The CPU1 OTP memory contains a 256-bit value that is made up of both pseudo-random and sequential parts. This value may be used as a seed for code encryption. The starting address of the value is 0x703C0. The first 192 bits are pseudo-random, the next 32 bits are sequential, and the last 32 bits are a checksum value. The degree of randomness is not specified.



Copyright © 2024 Texas Instruments Incorporated

## 3.14 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 may 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 it), there isdifference in how the application behaves with the debugger and without.

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

## 3.15 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 3-19. The formula to compute delay between subsequent writes:

**Delay (in SYSCLK cycles) = 3 × ($F_{SYSCLK}$ ÷ $F_{INTOSC1}$) + 9**

For Example - For SYSCLK = 100MHz

**Delay (in SYSCLK cycles) = 3 × (100MHz ÷ 10MHz) + 9 = 39 SYSCLK cycles**

**Table 3-19. System Control Registers Impacted**

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



Copyright © 2024 Texas Instruments Incorporated

## 3.16 Software

### *3.16.1 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 C2000Ware Examples.

#### 3.16.1.1 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 (200Mhz). 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 200Mhz 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

#### 3.16.1.2 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 GPIO73 using an oscilloscope.

### *3.16.2 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 C2000Ware Examples.

#### 3.16.2.1 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*



Copyright © 2024 Texas Instruments Incorporated

- cpuTimer0IntCount
- cpuTimer1IntCount
- cpuTimer2IntCount

#### 3.16.2.2 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.

This example project has support for migration across our C2000 device families. If you are wanting to build this project from launchpad or controlCARD, please specify in the .syscfg file the board you're using. At any time you can select another device to migrate this example. *External Connections*

- None

*Watch Variables*

- cpuTimer0IntCount
- cpuTimer1IntCount
- cpuTimer2IntCount

### *3.16.3 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 C2000Ware Examples.

### *3.16.4 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 C2000Ware Examples.

#### 3.16.4.1 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



Copyright © 2024 Texas Instruments Incorporated

#### 3.16.4.2 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

#### 3.16.4.3 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.



Copyright © 2024 Texas Instruments Incorporated

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

*External Connections*

- None

*Watch Variables*

- traceISR - shows the order in which ISRs are executed.

#### 3.16.4.4 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



Copyright © 2024 Texas Instruments Incorporated

### 3.16.5 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 C2000Ware Examples.

### 3.16.6 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.

#### 3.16.6.1 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



Copyright © 2024 Texas Instruments Incorporated

## 3.17 System Control Registers

### 3.17.1 System Control Base Addresses

**Table 3-20. System Control Base Address Table**

| Device Registers | Register Name | Start Address | End Address |
|---|---|---|---|
| CpuTimer0Regs | CPUTIMER_REGS | 0x0000_0C00 | 0x0000_0C07 |
| CpuTimer1Regs | CPUTIMER_REGS | 0x0000_0C08 | 0x0000_0C0F |
| CpuTimer2Regs | CPUTIMER_REGS | 0x0000_0C10 | 0x0000_0C17 |
| PieCtrlRegs | PIE_CTRL_REGS | 0x0000_0CE0 | 0x0000_0CFF |
| WdRegs | WD_REGS | 0x0000_7000 | 0x0000_703F |
| NmiIntruptRegs | NMI_INTRUPT_REGS | 0x0000_7060 | 0x0000_706F |
| XintRegs | XINT_REGS | 0x0000_7070 | 0x0000_707F |
| SyncSocRegs | SYNC_SOC_REGS | 0x0000_7940 | 0x0000_794F |
| DmaClaSrcSelRegs | DMA_CLA_SRC_SEL_REGS | 0x0000_7980 | 0x0000_79BF |
| FlashPumpSemaphoreRegs | FLASH_PUMP_SEMAPHORE_REGS | 0x0005_0024 | 0x0005_0025 |
| DevCfgRegs(1) | DEV_CFG_REGS | 0x0005_D000 | 0x0005_D17F |
| ClkCfgRegs | CLK_CFG_REGS | 0x0005_D200 | 0x0005_D2FF |
| CpuSysRegs | CPU_SYS_REGS | 0x0005_D300 | 0x0005_D3FF |
| RomPrefetchRegs(1) | ROM_PREFETCH_REGS | 0x0005_E608 | 0x0005_E609 |
| DcsmZ1Regs | DCSM_Z1_REGS | 0x0005_F000 | 0x0005_F02F |
| DcsmZ2Regs | DCSM_Z2_REGS | 0x0005_F040 | 0x0005_F05F |
| DcsmCommonRegs | DCSM_COMMON_REGS | 0x0005_F070 | 0x0005_F07F |
| MemCfgRegs | MEM_CFG_REGS | 0x0005_F400 | 0x0005_F47F |
| AccessProtectionRegs | ACCESS_PROTECTION_REGS | 0x0005_F4C0 | 0x0005_F4FF |
| MemoryErrorRegs | MEMORY_ERROR_REGS | 0x0005_F500 | 0x0005_F53F |
| RomWaitStateRegs(1) | ROM_WAIT_STATE_REGS | 0x0005_F540 | 0x0005_F541 |
| Flash0CtrlRegs | FLASH_CTRL_REGS | 0x0005_F800 | 0x0005_FAFF |
| Flash0EccRegs | FLASH_ECC_REGS | 0x0005_FB00 | 0x0005_FB3F |
| CpuIdRegs | CPU_ID_REGS | 0x0007_026D | 0x0007_026E |
| UidRegs | UID_REGS | 0x0007_03C0 | 0x0007_03CF |
| DcsmZ1OTPRegs | DCSM_Z1_OTP | 0x0007_8000 | 0x0007_81FF |
| DcsmZ2OTPRegs | DCSM_Z2_OTP | 0x0007_8200 | 0x0007_83FF |

(1) Only available on CPU1.



Copyright © 2024 Texas Instruments Incorporated

### 3.17.2 CPUTIMER_REGS Registers

Table 3-21 lists the memory-mapped registers for the CPUTIMER_REGS registers. All register offset addresses not listed in Table 3-21 should be considered as reserved locations and the register contents should not be modified.

**Table 3-21. CPUTIMER_REGS Registers**

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

Complex bit access types are encoded to fit into small table cells. Table 3-22 shows the codes that are used for access types in this section.

**Table 3-22. CPUTIMER_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| Write Type | | |
| W | W | Write |
| W1C | W<br>1C | Write<br>1 to clear |
| Reset or Default Value | | |
| *-n* | | Value after reset or the default value |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.2.1 TIM Register (Offset = 0h) [Reset = 0000FFFFh]

TIM is shown in Figure 3-26 and described in Table 3-23.

Return to the Summary Table.

CPU-Timer, Counter Register

**Figure 3-26. 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 3-23. TIM Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | MSW | R/W | 0h | CPU-Timer Counter Registers<br>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.<br>Reset type: SYSRSn |
| 15-0 | LSW | R/W | FFFFh | CPU-Timer Counter Registers<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.2.2 PRD Register (Offset = 2h) [Reset = 0000FFFFh]

PRD is shown in Figure 3-27 and described in Table 3-24.

Return to the Summary Table.

CPU-Timer, Period Register

**Figure 3-27. 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 3-24. PRD Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | MSW | R/W | 0h | CPU-Timer Period Registers<br>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).<br>Reset type: SYSRSn |
| 15-0 | LSW | R/W | FFFFh | CPU-Timer Period Registers<br>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).<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.2.3 TCR Register (Offset = 4h) [Reset = 0001h]

TCR is shown in Figure 3-28 and described in Table 3-25.

Return to the Summary Table.

CPU-Timer, Control Register

**Figure 3-28. 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 3-25. TCR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | TIF | R/W1C | 0h | CPU-Timer Overflow Flag.<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = The CPU-Timer has not decremented to zero.<br>Writes of 0 are ignored.<br>1h (R/W) = This flag gets set when the CPU-timer decrements to zero.<br>Writing a 1 to this bit clears the flag. |
| 14 | TIE | R/W | 0h | CPU-Timer Interrupt Enable.<br>Reset type: SYSRSn<br>0h (R/W) = The CPU-Timer interrupt is disabled.<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = Stop after the next decrement of the TIMH:TIM (hard stop)<br>(SOFT bit controls the emulation behavior)<br>1h (R/W) = Free Run<br>(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.<br>Reset type: SYSRSn<br>0h (R/W) = Stop after the next decrement of the TIMH:TIM (hard stop).<br>(ONLY if FREE=0, if FREE=1 this bit is don't care)<br>1h (R/W) = Stop after the TIMH:TIM decrements to 0 (soft stop)<br>In the SOFT STOP mode, the timer generates an interrupt before shutting down (since reaching 0 is the interrupt causing condition).<br>(ONLY if FREE=0, if FREE=1 this bit is don't care) |
| 9-6 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-25. TCR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 5 | TRB | R/W | 0h | Timer reload<br>Reset type: SYSRSn<br>0h (R/W) = The TRB bit is always read as zero. Writes of 0 are ignored.<br>1h (R/W) = When you write a 1 to TRB, the TIMH:TIM is loaded with the value in the PRDH:PRD,<br>and the prescaler counter (PSCH:PSC) is loaded with the value in the timer dividedown<br>register (TDDRH:TDDR). |
| 4 | TSS | R/W | 0h | CPU-Timer stop status bit.<br>TSS is a 1-bit flag that stops or starts the CPU-timer.<br>Reset type: SYSRSn<br>0h (R/W) = Reads of 0 indicate the CPU-timer is running.<br>To start or restart the CPU-timer, set TSS to 0. At reset, TSS is cleared to 0 and the<br>CPU-timer immediately starts.<br>1h (R/W) = Reads of 1 indicate that the CPU-timer is stopped.<br>To stop the CPU-timer, set TSS to 1. |
| 3-0 | RESERVED | R | 1h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.2.4 TPR Register (Offset = 6h) [Reset = 0000h]

TPR is shown in Figure 3-29 and described in Table 3-26.

Return to the Summary Table.

CPU-Timer, Prescale Register

**Figure 3-29. 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 3-26. TPR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | PSC | R | 0h | CPU-Timer Prescale Counter.<br>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.<br>Reset type: SYSRSn |
| 7-0 | TDDR | R/W | 0h | CPU-Timer Divide-Down.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.2.5 TPRH Register (Offset = 7h) [Reset = 0000h]

TPRH is shown in Figure 3-30 and described in Table 3-27.

Return to the Summary Table.

CPU-Timer, Prescale Register High

**Figure 3-30. 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 3-27. TPRH Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | PSCH | R | 0h | See description of TIMERxTPR.<br>Reset type: SYSRSn |
| 7-0 | TDDRH | R/W | 0h | See description of TIMERxTPR.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.3 PIE_CTRL_REGS Registers

Table 3-28 lists the memory-mapped registers for the PIE_CTRL_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. PIE_CTRL_REGS Registers**

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

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. PIE_CTRL_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| R-0 | R<br>-0 | Read<br>Returns 0s |
| Write Type | | |
| W | W | Write |
| W1S | W<br>1S | Write<br>1 to set |
| Reset or Default Value | | |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-29. PIE_CTRL_REGS Access Type Codes (continued)**

| Access Type | Code | Description |
|---|---|---|
| *-n* | | Value after reset or the default value |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.3.1 PIECTRL Register (Offset = 0h) [Reset = 0000h]

PIECTRL is shown in Figure 3-31 and described in Table 3-30.

Return to the Summary Table.

ePIE Control Register

**Figure 3-31. 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 3-30. 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.<br>Note: When a NMI is serviced, the PIEVECT bit-field does not reflect the vector as it does for other interrupts.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.3.2 PIEACK Register (Offset = 1h) [Reset = 0000h]

PIEACK is shown in Figure 3-32 and described in Table 3-31.

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 3-32. PIEACK Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | ACK12 | ACK11 | ACK10 | ACK9 |
| R-0-0h | | | | R/W1S-0h | R/W1S-0h | R/W1S-0h | R/W1S-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ACK8 | ACK7 | ACK6 | ACK5 | ACK4 | ACK3 | ACK2 | ACK1 |
| R/W1S-0h | R/W1S-0h | R/W1S-0h | R/W1S-0h | R/W1S-0h | R/W1S-0h | R/W1S-0h | R/W1S-0h |

**Table 3-31. 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<br>Reset type: SYSRSn |
| 10 | ACK11 | R/W1S | 0h | Acknowledge PIE Interrupt Group 11<br>Reset type: SYSRSn |
| 9 | ACK10 | R/W1S | 0h | Acknowledge PIE Interrupt Group 10<br>Reset type: SYSRSn |
| 8 | ACK9 | R/W1S | 0h | Acknowledge PIE Interrupt Group 9<br>Reset type: SYSRSn |
| 7 | ACK8 | R/W1S | 0h | Acknowledge PIE Interrupt Group 8<br>Reset type: SYSRSn |
| 6 | ACK7 | R/W1S | 0h | Acknowledge PIE Interrupt Group 7<br>Reset type: SYSRSn |
| 5 | ACK6 | R/W1S | 0h | Acknowledge PIE Interrupt Group 6<br>Reset type: SYSRSn |
| 4 | ACK5 | R/W1S | 0h | Acknowledge PIE Interrupt Group 5<br>Reset type: SYSRSn |
| 3 | ACK4 | R/W1S | 0h | Acknowledge PIE Interrupt Group 4<br>Reset type: SYSRSn |
| 2 | ACK3 | R/W1S | 0h | Acknowledge PIE Interrupt Group 3<br>Reset type: SYSRSn |
| 1 | ACK2 | R/W1S | 0h | Acknowledge PIE Interrupt Group 2<br>Reset type: SYSRSn |
| 0 | ACK1 | R/W1S | 0h | Acknowledge PIE Interrupt Group 1<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.3.3 PIEIER1 Register (Offset = 2h) [Reset = 0000h]

PIEIER1 is shown in Figure 3-33 and described in Table 3-32.

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 3-33. PIEIER1 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| INTx16 | INTx15 | INTx14 | INTx13 | INTx12 | INTx11 | INTx10 | INTx9 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTx8 | INTx7 | INTx6 | INTx5 | INTx4 | INTx3 | INTx2 | INTx1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-32. PIEIER1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | INTx16 | R/W | 0h | Enable for Interrupt 1.16<br>Reset type: SYSRSn |
| 14 | INTx15 | R/W | 0h | Enable for Interrupt 1.15<br>Reset type: SYSRSn |
| 13 | INTx14 | R/W | 0h | Enable for Interrupt 1.14<br>Reset type: SYSRSn |
| 12 | INTx13 | R/W | 0h | Enable for Interrupt 1.13<br>Reset type: SYSRSn |
| 11 | INTx12 | R/W | 0h | Enable for Interrupt 1.12<br>Reset type: SYSRSn |
| 10 | INTx11 | R/W | 0h | Enable for Interrupt 1.11<br>Reset type: SYSRSn |
| 9 | INTx10 | R/W | 0h | Enable for Interrupt 1.10<br>Reset type: SYSRSn |
| 8 | INTx9 | R/W | 0h | Enable for Interrupt 1.9<br>Reset type: SYSRSn |
| 7 | INTx8 | R/W | 0h | Enable for Interrupt 1.8<br>Reset type: SYSRSn |
| 6 | INTx7 | R/W | 0h | Enable for Interrupt 1.7<br>Reset type: SYSRSn |
| 5 | INTx6 | R/W | 0h | Enable for Interrupt 1.6<br>Reset type: SYSRSn |
| 4 | INTx5 | R/W | 0h | Enable for Interrupt 1.5<br>Reset type: SYSRSn |
| 3 | INTx4 | R/W | 0h | Enable for Interrupt 1.4<br>Reset type: SYSRSn |
| 2 | INTx3 | R/W | 0h | Enable for Interrupt 1.3<br>Reset type: SYSRSn |
| 1 | INTx2 | R/W | 0h | Enable for Interrupt 1.2<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-32. PIEIER1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | INTx1 | R/W | 0h | Enable for Interrupt 1.1<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.3.4 PIEIFR1 Register (Offset = 3h) [Reset = 0000h]

PIEIFR1 is shown in Figure 3-34 and described in Table 3-33.

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 3-34. PIEIFR1 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| INTx16 | INTx15 | INTx14 | INTx13 | INTx12 | INTx11 | INTx10 | INTx9 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTx8 | INTx7 | INTx6 | INTx5 | INTx4 | INTx3 | INTx2 | INTx1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-33. PIEIFR1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | INTx16 | R/W | 0h | Flag for Interrupt 1.16<br>Reset type: SYSRSn |
| 14 | INTx15 | R/W | 0h | Flag for Interrupt 1.15<br>Reset type: SYSRSn |
| 13 | INTx14 | R/W | 0h | Flag for Interrupt 1.14<br>Reset type: SYSRSn |
| 12 | INTx13 | R/W | 0h | Flag for Interrupt 1.13<br>Reset type: SYSRSn |
| 11 | INTx12 | R/W | 0h | Flag for Interrupt 1.12<br>Reset type: SYSRSn |
| 10 | INTx11 | R/W | 0h | Flag for Interrupt 1.11<br>Reset type: SYSRSn |
| 9 | INTx10 | R/W | 0h | Flag for Interrupt 1.10<br>Reset type: SYSRSn |
| 8 | INTx9 | R/W | 0h | Flag for Interrupt 1.9<br>Reset type: SYSRSn |
| 7 | INTx8 | R/W | 0h | Flag for Interrupt 1.8<br>Reset type: SYSRSn |
| 6 | INTx7 | R/W | 0h | Flag for Interrupt 1.7<br>Reset type: SYSRSn |
| 5 | INTx6 | R/W | 0h | Flag for Interrupt 1.6<br>Reset type: SYSRSn |
| 4 | INTx5 | R/W | 0h | Flag for Interrupt 1.5<br>Reset type: SYSRSn |
| 3 | INTx4 | R/W | 0h | Flag for Interrupt 1.4<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-33. PIEIFR1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 2 | INTx3 | R/W | 0h | Flag for Interrupt 1.3<br>Reset type: SYSRSn |
| 1 | INTx2 | R/W | 0h | Flag for Interrupt 1.2<br>Reset type: SYSRSn |
| 0 | INTx1 | R/W | 0h | Flag for Interrupt 1.1<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.3.5 PIEIER2 Register (Offset = 4h) [Reset = 0000h]

PIEIER2 is shown in Figure 3-35 and described in Table 3-34.

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 3-35. PIEIER2 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| INTx16 | INTx15 | INTx14 | INTx13 | INTx12 | INTx11 | INTx10 | INTx9 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTx8 | INTx7 | INTx6 | INTx5 | INTx4 | INTx3 | INTx2 | INTx1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-34. PIEIER2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | INTx16 | R/W | 0h | Enable for Interrupt 2.16<br>Reset type: SYSRSn |
| 14 | INTx15 | R/W | 0h | Enable for Interrupt 2.15<br>Reset type: SYSRSn |
| 13 | INTx14 | R/W | 0h | Enable for Interrupt 2.14<br>Reset type: SYSRSn |
| 12 | INTx13 | R/W | 0h | Enable for Interrupt 2.13<br>Reset type: SYSRSn |
| 11 | INTx12 | R/W | 0h | Enable for Interrupt 2.12<br>Reset type: SYSRSn |
| 10 | INTx11 | R/W | 0h | Enable for Interrupt 2.11<br>Reset type: SYSRSn |
| 9 | INTx10 | R/W | 0h | Enable for Interrupt 2.10<br>Reset type: SYSRSn |
| 8 | INTx9 | R/W | 0h | Enable for Interrupt 2.9<br>Reset type: SYSRSn |
| 7 | INTx8 | R/W | 0h | Enable for Interrupt 2.8<br>Reset type: SYSRSn |
| 6 | INTx7 | R/W | 0h | Enable for Interrupt 2.7<br>Reset type: SYSRSn |
| 5 | INTx6 | R/W | 0h | Enable for Interrupt 2.6<br>Reset type: SYSRSn |
| 4 | INTx5 | R/W | 0h | Enable for Interrupt 2.5<br>Reset type: SYSRSn |
| 3 | INTx4 | R/W | 0h | Enable for Interrupt 2.4<br>Reset type: SYSRSn |
| 2 | INTx3 | R/W | 0h | Enable for Interrupt 2.3<br>Reset type: SYSRSn |
| 1 | INTx2 | R/W | 0h | Enable for Interrupt 2.2<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-34. PIEIER2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | INTx1 | R/W | 0h | Enable for Interrupt 2.1<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.3.6 PIEIFR2 Register (Offset = 5h) [Reset = 0000h]

PIEIFR2 is shown in Figure 3-36 and described in Table 3-35.

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 3-36. PIEIFR2 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| INTx16 | INTx15 | INTx14 | INTx13 | INTx12 | INTx11 | INTx10 | INTx9 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTx8 | INTx7 | INTx6 | INTx5 | INTx4 | INTx3 | INTx2 | INTx1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-35. PIEIFR2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | INTx16 | R/W | 0h | Flag for Interrupt 2.16<br>Reset type: SYSRSn |
| 14 | INTx15 | R/W | 0h | Flag for Interrupt 2.15<br>Reset type: SYSRSn |
| 13 | INTx14 | R/W | 0h | Flag for Interrupt 2.14<br>Reset type: SYSRSn |
| 12 | INTx13 | R/W | 0h | Flag for Interrupt 2.13<br>Reset type: SYSRSn |
| 11 | INTx12 | R/W | 0h | Flag for Interrupt 2.12<br>Reset type: SYSRSn |
| 10 | INTx11 | R/W | 0h | Flag for Interrupt 2.11<br>Reset type: SYSRSn |
| 9 | INTx10 | R/W | 0h | Flag for Interrupt 2.10<br>Reset type: SYSRSn |
| 8 | INTx9 | R/W | 0h | Flag for Interrupt 2.9<br>Reset type: SYSRSn |
| 7 | INTx8 | R/W | 0h | Flag for Interrupt 2.8<br>Reset type: SYSRSn |
| 6 | INTx7 | R/W | 0h | Flag for Interrupt 2.7<br>Reset type: SYSRSn |
| 5 | INTx6 | R/W | 0h | Flag for Interrupt 2.6<br>Reset type: SYSRSn |
| 4 | INTx5 | R/W | 0h | Flag for Interrupt 2.5<br>Reset type: SYSRSn |
| 3 | INTx4 | R/W | 0h | Flag for Interrupt 2.4<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-35. PIEIFR2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 2 | INTx3 | R/W | 0h | Flag for Interrupt 2.3<br>Reset type: SYSRSn |
| 1 | INTx2 | R/W | 0h | Flag for Interrupt 2.2<br>Reset type: SYSRSn |
| 0 | INTx1 | R/W | 0h | Flag for Interrupt 2.1<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.3.7 PIEIER3 Register (Offset = 6h) [Reset = 0000h]

PIEIER3 is shown in Figure 3-37 and described in Table 3-36.

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 3-37. PIEIER3 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| INTx16 | INTx15 | INTx14 | INTx13 | INTx12 | INTx11 | INTx10 | INTx9 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTx8 | INTx7 | INTx6 | INTx5 | INTx4 | INTx3 | INTx2 | INTx1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-36. PIEIER3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | INTx16 | R/W | 0h | Enable for Interrupt 3.16<br>Reset type: SYSRSn |
| 14 | INTx15 | R/W | 0h | Enable for Interrupt 3.15<br>Reset type: SYSRSn |
| 13 | INTx14 | R/W | 0h | Enable for Interrupt 3.14<br>Reset type: SYSRSn |
| 12 | INTx13 | R/W | 0h | Enable for Interrupt 3.13<br>Reset type: SYSRSn |
| 11 | INTx12 | R/W | 0h | Enable for Interrupt 3.12<br>Reset type: SYSRSn |
| 10 | INTx11 | R/W | 0h | Enable for Interrupt 3.11<br>Reset type: SYSRSn |
| 9 | INTx10 | R/W | 0h | Enable for Interrupt 3.10<br>Reset type: SYSRSn |
| 8 | INTx9 | R/W | 0h | Enable for Interrupt 3.9<br>Reset type: SYSRSn |
| 7 | INTx8 | R/W | 0h | Enable for Interrupt 3.8<br>Reset type: SYSRSn |
| 6 | INTx7 | R/W | 0h | Enable for Interrupt 3.7<br>Reset type: SYSRSn |
| 5 | INTx6 | R/W | 0h | Enable for Interrupt 3.6<br>Reset type: SYSRSn |
| 4 | INTx5 | R/W | 0h | Enable for Interrupt 3.5<br>Reset type: SYSRSn |
| 3 | INTx4 | R/W | 0h | Enable for Interrupt 3.4<br>Reset type: SYSRSn |
| 2 | INTx3 | R/W | 0h | Enable for Interrupt 3.3<br>Reset type: SYSRSn |
| 1 | INTx2 | R/W | 0h | Enable for Interrupt 3.2<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-36. PIEIER3 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | INTx1 | R/W | 0h | Enable for Interrupt 3.1<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.3.8 PIEIFR3 Register (Offset = 7h) [Reset = 0000h]

PIEIFR3 is shown in Figure 3-38 and described in Table 3-37.

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 3-38. PIEIFR3 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| INTx16 | INTx15 | INTx14 | INTx13 | INTx12 | INTx11 | INTx10 | INTx9 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTx8 | INTx7 | INTx6 | INTx5 | INTx4 | INTx3 | INTx2 | INTx1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-37. PIEIFR3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | INTx16 | R/W | 0h | Flag for Interrupt 3.16<br>Reset type: SYSRSn |
| 14 | INTx15 | R/W | 0h | Flag for Interrupt 3.15<br>Reset type: SYSRSn |
| 13 | INTx14 | R/W | 0h | Flag for Interrupt 3.14<br>Reset type: SYSRSn |
| 12 | INTx13 | R/W | 0h | Flag for Interrupt 3.13<br>Reset type: SYSRSn |
| 11 | INTx12 | R/W | 0h | Flag for Interrupt 3.12<br>Reset type: SYSRSn |
| 10 | INTx11 | R/W | 0h | Flag for Interrupt 3.11<br>Reset type: SYSRSn |
| 9 | INTx10 | R/W | 0h | Flag for Interrupt 3.10<br>Reset type: SYSRSn |
| 8 | INTx9 | R/W | 0h | Flag for Interrupt 3.9<br>Reset type: SYSRSn |
| 7 | INTx8 | R/W | 0h | Flag for Interrupt 3.8<br>Reset type: SYSRSn |
| 6 | INTx7 | R/W | 0h | Flag for Interrupt 3.7<br>Reset type: SYSRSn |
| 5 | INTx6 | R/W | 0h | Flag for Interrupt 3.6<br>Reset type: SYSRSn |
| 4 | INTx5 | R/W | 0h | Flag for Interrupt 3.5<br>Reset type: SYSRSn |
| 3 | INTx4 | R/W | 0h | Flag for Interrupt 3.4<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-37. PIEIFR3 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 2 | INTx3 | R/W | 0h | Flag for Interrupt 3.3<br>Reset type: SYSRSn |
| 1 | INTx2 | R/W | 0h | Flag for Interrupt 3.2<br>Reset type: SYSRSn |
| 0 | INTx1 | R/W | 0h | Flag for Interrupt 3.1<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.3.9 PIEIER4 Register (Offset = 8h) [Reset = 0000h]

PIEIER4 is shown in Figure 3-39 and described in Table 3-38.

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 3-39. PIEIER4 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| INTx16 | INTx15 | INTx14 | INTx13 | INTx12 | INTx11 | INTx10 | INTx9 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTx8 | INTx7 | INTx6 | INTx5 | INTx4 | INTx3 | INTx2 | INTx1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-38. PIEIER4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | INTx16 | R/W | 0h | Enable for Interrupt 4.16<br>Reset type: SYSRSn |
| 14 | INTx15 | R/W | 0h | Enable for Interrupt 4.15<br>Reset type: SYSRSn |
| 13 | INTx14 | R/W | 0h | Enable for Interrupt 4.14<br>Reset type: SYSRSn |
| 12 | INTx13 | R/W | 0h | Enable for Interrupt 4.13<br>Reset type: SYSRSn |
| 11 | INTx12 | R/W | 0h | Enable for Interrupt 4.12<br>Reset type: SYSRSn |
| 10 | INTx11 | R/W | 0h | Enable for Interrupt 4.11<br>Reset type: SYSRSn |
| 9 | INTx10 | R/W | 0h | Enable for Interrupt 4.10<br>Reset type: SYSRSn |
| 8 | INTx9 | R/W | 0h | Enable for Interrupt 4.9<br>Reset type: SYSRSn |
| 7 | INTx8 | R/W | 0h | Enable for Interrupt 4.8<br>Reset type: SYSRSn |
| 6 | INTx7 | R/W | 0h | Enable for Interrupt 4.7<br>Reset type: SYSRSn |
| 5 | INTx6 | R/W | 0h | Enable for Interrupt 4.6<br>Reset type: SYSRSn |
| 4 | INTx5 | R/W | 0h | Enable for Interrupt 4.5<br>Reset type: SYSRSn |
| 3 | INTx4 | R/W | 0h | Enable for Interrupt 4.4<br>Reset type: SYSRSn |
| 2 | INTx3 | R/W | 0h | Enable for Interrupt 4.3<br>Reset type: SYSRSn |
| 1 | INTx2 | R/W | 0h | Enable for Interrupt 4.2<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-38. PIEIER4 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | INTx1 | R/W | 0h | Enable for Interrupt 4.1<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.3.10 PIEIFR4 Register (Offset = 9h) [Reset = 0000h]

PIEIFR4 is shown in Figure 3-40 and described in Table 3-39.

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 3-40. PIEIFR4 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| INTx16 | INTx15 | INTx14 | INTx13 | INTx12 | INTx11 | INTx10 | INTx9 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTx8 | INTx7 | INTx6 | INTx5 | INTx4 | INTx3 | INTx2 | INTx1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-39. PIEIFR4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | INTx16 | R/W | 0h | Flag for Interrupt 4.16<br>Reset type: SYSRSn |
| 14 | INTx15 | R/W | 0h | Flag for Interrupt 4.15<br>Reset type: SYSRSn |
| 13 | INTx14 | R/W | 0h | Flag for Interrupt 4.14<br>Reset type: SYSRSn |
| 12 | INTx13 | R/W | 0h | Flag for Interrupt 4.13<br>Reset type: SYSRSn |
| 11 | INTx12 | R/W | 0h | Flag for Interrupt 4.12<br>Reset type: SYSRSn |
| 10 | INTx11 | R/W | 0h | Flag for Interrupt 4.11<br>Reset type: SYSRSn |
| 9 | INTx10 | R/W | 0h | Flag for Interrupt 4.10<br>Reset type: SYSRSn |
| 8 | INTx9 | R/W | 0h | Flag for Interrupt 4.9<br>Reset type: SYSRSn |
| 7 | INTx8 | R/W | 0h | Flag for Interrupt 4.8<br>Reset type: SYSRSn |
| 6 | INTx7 | R/W | 0h | Flag for Interrupt 4.7<br>Reset type: SYSRSn |
| 5 | INTx6 | R/W | 0h | Flag for Interrupt 4.6<br>Reset type: SYSRSn |
| 4 | INTx5 | R/W | 0h | Flag for Interrupt 4.5<br>Reset type: SYSRSn |
| 3 | INTx4 | R/W | 0h | Flag for Interrupt 4.4<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-39. PIEIFR4 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 2 | INTx3 | R/W | 0h | Flag for Interrupt 4.3<br>Reset type: SYSRSn |
| 1 | INTx2 | R/W | 0h | Flag for Interrupt 4.2<br>Reset type: SYSRSn |
| 0 | INTx1 | R/W | 0h | Flag for Interrupt 4.1<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.3.11 PIEIER5 Register (Offset = Ah) [Reset = 0000h]

PIEIER5 is shown in Figure 3-41 and described in Table 3-40.

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 3-41. PIEIER5 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| INTx16 | INTx15 | INTx14 | INTx13 | INTx12 | INTx11 | INTx10 | INTx9 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTx8 | INTx7 | INTx6 | INTx5 | INTx4 | INTx3 | INTx2 | INTx1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-40. PIEIER5 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | INTx16 | R/W | 0h | Enable for Interrupt 5.16<br>Reset type: SYSRSn |
| 14 | INTx15 | R/W | 0h | Enable for Interrupt 5.15<br>Reset type: SYSRSn |
| 13 | INTx14 | R/W | 0h | Enable for Interrupt 5.14<br>Reset type: SYSRSn |
| 12 | INTx13 | R/W | 0h | Enable for Interrupt 5.13<br>Reset type: SYSRSn |
| 11 | INTx12 | R/W | 0h | Enable for Interrupt 5.12<br>Reset type: SYSRSn |
| 10 | INTx11 | R/W | 0h | Enable for Interrupt 5.11<br>Reset type: SYSRSn |
| 9 | INTx10 | R/W | 0h | Enable for Interrupt 5.10<br>Reset type: SYSRSn |
| 8 | INTx9 | R/W | 0h | Enable for Interrupt 5.9<br>Reset type: SYSRSn |
| 7 | INTx8 | R/W | 0h | Enable for Interrupt 5.8<br>Reset type: SYSRSn |
| 6 | INTx7 | R/W | 0h | Enable for Interrupt 5.7<br>Reset type: SYSRSn |
| 5 | INTx6 | R/W | 0h | Enable for Interrupt 5.6<br>Reset type: SYSRSn |
| 4 | INTx5 | R/W | 0h | Enable for Interrupt 5.5<br>Reset type: SYSRSn |
| 3 | INTx4 | R/W | 0h | Enable for Interrupt 5.4<br>Reset type: SYSRSn |
| 2 | INTx3 | R/W | 0h | Enable for Interrupt 5.3<br>Reset type: SYSRSn |
| 1 | INTx2 | R/W | 0h | Enable for Interrupt 5.2<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-40. PIEIER5 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | INTx1 | R/W | 0h | Enable for Interrupt 5.1<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.3.12 PIEIFR5 Register (Offset = Bh) [Reset = 0000h]

PIEIFR5 is shown in Figure 3-42 and described in Table 3-41.

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 3-42. PIEIFR5 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| INTx16 | INTx15 | INTx14 | INTx13 | INTx12 | INTx11 | INTx10 | INTx9 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTx8 | INTx7 | INTx6 | INTx5 | INTx4 | INTx3 | INTx2 | INTx1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-41. PIEIFR5 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | INTx16 | R/W | 0h | Flag for Interrupt 5.16<br>Reset type: SYSRSn |
| 14 | INTx15 | R/W | 0h | Flag for Interrupt 5.15<br>Reset type: SYSRSn |
| 13 | INTx14 | R/W | 0h | Flag for Interrupt 5.14<br>Reset type: SYSRSn |
| 12 | INTx13 | R/W | 0h | Flag for Interrupt 5.13<br>Reset type: SYSRSn |
| 11 | INTx12 | R/W | 0h | Flag for Interrupt 5.12<br>Reset type: SYSRSn |
| 10 | INTx11 | R/W | 0h | Flag for Interrupt 5.11<br>Reset type: SYSRSn |
| 9 | INTx10 | R/W | 0h | Flag for Interrupt 5.10<br>Reset type: SYSRSn |
| 8 | INTx9 | R/W | 0h | Flag for Interrupt 5.9<br>Reset type: SYSRSn |
| 7 | INTx8 | R/W | 0h | Flag for Interrupt 5.8<br>Reset type: SYSRSn |
| 6 | INTx7 | R/W | 0h | Flag for Interrupt 5.7<br>Reset type: SYSRSn |
| 5 | INTx6 | R/W | 0h | Flag for Interrupt 5.6<br>Reset type: SYSRSn |
| 4 | INTx5 | R/W | 0h | Flag for Interrupt 5.5<br>Reset type: SYSRSn |
| 3 | INTx4 | R/W | 0h | Flag for Interrupt 5.4<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-41. PIEIFR5 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 2 | INTx3 | R/W | 0h | Flag for Interrupt 5.3<br>Reset type: SYSRSn |
| 1 | INTx2 | R/W | 0h | Flag for Interrupt 5.2<br>Reset type: SYSRSn |
| 0 | INTx1 | R/W | 0h | Flag for Interrupt 5.1<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.3.13 PIEIER6 Register (Offset = Ch) [Reset = 0000h]

PIEIER6 is shown in Figure 3-43 and described in Table 3-42.

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 3-43. PIEIER6 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| INTx16 | INTx15 | INTx14 | INTx13 | INTx12 | INTx11 | INTx10 | INTx9 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTx8 | INTx7 | INTx6 | INTx5 | INTx4 | INTx3 | INTx2 | INTx1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-42. PIEIER6 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | INTx16 | R/W | 0h | Enable for Interrupt 6.16<br>Reset type: SYSRSn |
| 14 | INTx15 | R/W | 0h | Enable for Interrupt 6.15<br>Reset type: SYSRSn |
| 13 | INTx14 | R/W | 0h | Enable for Interrupt 6.14<br>Reset type: SYSRSn |
| 12 | INTx13 | R/W | 0h | Enable for Interrupt 6.13<br>Reset type: SYSRSn |
| 11 | INTx12 | R/W | 0h | Enable for Interrupt 6.12<br>Reset type: SYSRSn |
| 10 | INTx11 | R/W | 0h | Enable for Interrupt 6.11<br>Reset type: SYSRSn |
| 9 | INTx10 | R/W | 0h | Enable for Interrupt 6.10<br>Reset type: SYSRSn |
| 8 | INTx9 | R/W | 0h | Enable for Interrupt 6.9<br>Reset type: SYSRSn |
| 7 | INTx8 | R/W | 0h | Enable for Interrupt 6.8<br>Reset type: SYSRSn |
| 6 | INTx7 | R/W | 0h | Enable for Interrupt 6.7<br>Reset type: SYSRSn |
| 5 | INTx6 | R/W | 0h | Enable for Interrupt 6.6<br>Reset type: SYSRSn |
| 4 | INTx5 | R/W | 0h | Enable for Interrupt 6.5<br>Reset type: SYSRSn |
| 3 | INTx4 | R/W | 0h | Enable for Interrupt 6.4<br>Reset type: SYSRSn |
| 2 | INTx3 | R/W | 0h | Enable for Interrupt 6.3<br>Reset type: SYSRSn |
| 1 | INTx2 | R/W | 0h | Enable for Interrupt 6.2<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-42. PIEIER6 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | INTx1 | R/W | 0h | Enable for Interrupt 6.1<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.3.14 PIEIFR6 Register (Offset = Dh) [Reset = 0000h]

PIEIFR6 is shown in Figure 3-44 and described in Table 3-43.

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 3-44. PIEIFR6 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| INTx16 | INTx15 | INTx14 | INTx13 | INTx12 | INTx11 | INTx10 | INTx9 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTx8 | INTx7 | INTx6 | INTx5 | INTx4 | INTx3 | INTx2 | INTx1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-43. PIEIFR6 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | INTx16 | R/W | 0h | Flag for Interrupt 6.16<br>Reset type: SYSRSn |
| 14 | INTx15 | R/W | 0h | Flag for Interrupt 6.15<br>Reset type: SYSRSn |
| 13 | INTx14 | R/W | 0h | Flag for Interrupt 6.14<br>Reset type: SYSRSn |
| 12 | INTx13 | R/W | 0h | Flag for Interrupt 6.13<br>Reset type: SYSRSn |
| 11 | INTx12 | R/W | 0h | Flag for Interrupt 6.12<br>Reset type: SYSRSn |
| 10 | INTx11 | R/W | 0h | Flag for Interrupt 6.11<br>Reset type: SYSRSn |
| 9 | INTx10 | R/W | 0h | Flag for Interrupt 6.10<br>Reset type: SYSRSn |
| 8 | INTx9 | R/W | 0h | Flag for Interrupt 6.9<br>Reset type: SYSRSn |
| 7 | INTx8 | R/W | 0h | Flag for Interrupt 6.8<br>Reset type: SYSRSn |
| 6 | INTx7 | R/W | 0h | Flag for Interrupt 6.7<br>Reset type: SYSRSn |
| 5 | INTx6 | R/W | 0h | Flag for Interrupt 6.6<br>Reset type: SYSRSn |
| 4 | INTx5 | R/W | 0h | Flag for Interrupt 6.5<br>Reset type: SYSRSn |
| 3 | INTx4 | R/W | 0h | Flag for Interrupt 6.4<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-43. PIEIFR6 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 2 | INTx3 | R/W | 0h | Flag for Interrupt 6.3<br>Reset type: SYSRSn |
| 1 | INTx2 | R/W | 0h | Flag for Interrupt 6.2<br>Reset type: SYSRSn |
| 0 | INTx1 | R/W | 0h | Flag for Interrupt 6.1<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.3.15 PIEIER7 Register (Offset = Eh) [Reset = 0000h]

PIEIER7 is shown in Figure 3-45 and described in Table 3-44.

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 3-45. PIEIER7 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| INTx16 | INTx15 | INTx14 | INTx13 | INTx12 | INTx11 | INTx10 | INTx9 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTx8 | INTx7 | INTx6 | INTx5 | INTx4 | INTx3 | INTx2 | INTx1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-44. PIEIER7 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | INTx16 | R/W | 0h | Enable for Interrupt 7.16<br>Reset type: SYSRSn |
| 14 | INTx15 | R/W | 0h | Enable for Interrupt 7.15<br>Reset type: SYSRSn |
| 13 | INTx14 | R/W | 0h | Enable for Interrupt 7.14<br>Reset type: SYSRSn |
| 12 | INTx13 | R/W | 0h | Enable for Interrupt 7.13<br>Reset type: SYSRSn |
| 11 | INTx12 | R/W | 0h | Enable for Interrupt 7.12<br>Reset type: SYSRSn |
| 10 | INTx11 | R/W | 0h | Enable for Interrupt 7.11<br>Reset type: SYSRSn |
| 9 | INTx10 | R/W | 0h | Enable for Interrupt 7.10<br>Reset type: SYSRSn |
| 8 | INTx9 | R/W | 0h | Enable for Interrupt 7.9<br>Reset type: SYSRSn |
| 7 | INTx8 | R/W | 0h | Enable for Interrupt 7.8<br>Reset type: SYSRSn |
| 6 | INTx7 | R/W | 0h | Enable for Interrupt 7.7<br>Reset type: SYSRSn |
| 5 | INTx6 | R/W | 0h | Enable for Interrupt 7.6<br>Reset type: SYSRSn |
| 4 | INTx5 | R/W | 0h | Enable for Interrupt 7.5<br>Reset type: SYSRSn |
| 3 | INTx4 | R/W | 0h | Enable for Interrupt 7.4<br>Reset type: SYSRSn |
| 2 | INTx3 | R/W | 0h | Enable for Interrupt 7.3<br>Reset type: SYSRSn |
| 1 | INTx2 | R/W | 0h | Enable for Interrupt 7.2<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-44. PIEIER7 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | INTx1 | R/W | 0h | Enable for Interrupt 7.1<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.3.16 PIEIFR7 Register (Offset = Fh) [Reset = 0000h]

PIEIFR7 is shown in Figure 3-46 and described in Table 3-45.

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 3-46. PIEIFR7 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| INTx16 | INTx15 | INTx14 | INTx13 | INTx12 | INTx11 | INTx10 | INTx9 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTx8 | INTx7 | INTx6 | INTx5 | INTx4 | INTx3 | INTx2 | INTx1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-45. PIEIFR7 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | INTx16 | R/W | 0h | Flag for Interrupt 7.16<br>Reset type: SYSRSn |
| 14 | INTx15 | R/W | 0h | Flag for Interrupt 7.15<br>Reset type: SYSRSn |
| 13 | INTx14 | R/W | 0h | Flag for Interrupt 7.14<br>Reset type: SYSRSn |
| 12 | INTx13 | R/W | 0h | Flag for Interrupt 7.13<br>Reset type: SYSRSn |
| 11 | INTx12 | R/W | 0h | Flag for Interrupt 7.12<br>Reset type: SYSRSn |
| 10 | INTx11 | R/W | 0h | Flag for Interrupt 7.11<br>Reset type: SYSRSn |
| 9 | INTx10 | R/W | 0h | Flag for Interrupt 7.10<br>Reset type: SYSRSn |
| 8 | INTx9 | R/W | 0h | Flag for Interrupt 7.9<br>Reset type: SYSRSn |
| 7 | INTx8 | R/W | 0h | Flag for Interrupt 7.8<br>Reset type: SYSRSn |
| 6 | INTx7 | R/W | 0h | Flag for Interrupt 7.7<br>Reset type: SYSRSn |
| 5 | INTx6 | R/W | 0h | Flag for Interrupt 7.6<br>Reset type: SYSRSn |
| 4 | INTx5 | R/W | 0h | Flag for Interrupt 7.5<br>Reset type: SYSRSn |
| 3 | INTx4 | R/W | 0h | Flag for Interrupt 7.4<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-45. PIEIFR7 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 2 | INTx3 | R/W | 0h | Flag for Interrupt 7.3<br>Reset type: SYSRSn |
| 1 | INTx2 | R/W | 0h | Flag for Interrupt 7.2<br>Reset type: SYSRSn |
| 0 | INTx1 | R/W | 0h | Flag for Interrupt 7.1<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.3.17 PIEIER8 Register (Offset = 10h) [Reset = 0000h]

PIEIER8 is shown in Figure 3-47 and described in Table 3-46.

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 3-47. PIEIER8 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| INTx16 | INTx15 | INTx14 | INTx13 | INTx12 | INTx11 | INTx10 | INTx9 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTx8 | INTx7 | INTx6 | INTx5 | INTx4 | INTx3 | INTx2 | INTx1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-46. PIEIER8 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | INTx16 | R/W | 0h | Enable for Interrupt 8.16<br>Reset type: SYSRSn |
| 14 | INTx15 | R/W | 0h | Enable for Interrupt 8.15<br>Reset type: SYSRSn |
| 13 | INTx14 | R/W | 0h | Enable for Interrupt 8.14<br>Reset type: SYSRSn |
| 12 | INTx13 | R/W | 0h | Enable for Interrupt 8.13<br>Reset type: SYSRSn |
| 11 | INTx12 | R/W | 0h | Enable for Interrupt 8.12<br>Reset type: SYSRSn |
| 10 | INTx11 | R/W | 0h | Enable for Interrupt 8.11<br>Reset type: SYSRSn |
| 9 | INTx10 | R/W | 0h | Enable for Interrupt 8.10<br>Reset type: SYSRSn |
| 8 | INTx9 | R/W | 0h | Enable for Interrupt 8.9<br>Reset type: SYSRSn |
| 7 | INTx8 | R/W | 0h | Enable for Interrupt 8.8<br>Reset type: SYSRSn |
| 6 | INTx7 | R/W | 0h | Enable for Interrupt 8.7<br>Reset type: SYSRSn |
| 5 | INTx6 | R/W | 0h | Enable for Interrupt 8.6<br>Reset type: SYSRSn |
| 4 | INTx5 | R/W | 0h | Enable for Interrupt 8.5<br>Reset type: SYSRSn |
| 3 | INTx4 | R/W | 0h | Enable for Interrupt 8.4<br>Reset type: SYSRSn |
| 2 | INTx3 | R/W | 0h | Enable for Interrupt 8.3<br>Reset type: SYSRSn |
| 1 | INTx2 | R/W | 0h | Enable for Interrupt 8.2<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-46. PIEIER8 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | INTx1 | R/W | 0h | Enable for Interrupt 8.1<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.3.18 PIEIFR8 Register (Offset = 11h) [Reset = 0000h]

PIEIFR8 is shown in Figure 3-48 and described in Table 3-47.

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 3-48. PIEIFR8 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| INTx16 | INTx15 | INTx14 | INTx13 | INTx12 | INTx11 | INTx10 | INTx9 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTx8 | INTx7 | INTx6 | INTx5 | INTx4 | INTx3 | INTx2 | INTx1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-47. PIEIFR8 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | INTx16 | R/W | 0h | Flag for Interrupt 8.16<br>Reset type: SYSRSn |
| 14 | INTx15 | R/W | 0h | Flag for Interrupt 8.15<br>Reset type: SYSRSn |
| 13 | INTx14 | R/W | 0h | Flag for Interrupt 8.14<br>Reset type: SYSRSn |
| 12 | INTx13 | R/W | 0h | Flag for Interrupt 8.13<br>Reset type: SYSRSn |
| 11 | INTx12 | R/W | 0h | Flag for Interrupt 8.12<br>Reset type: SYSRSn |
| 10 | INTx11 | R/W | 0h | Flag for Interrupt 8.11<br>Reset type: SYSRSn |
| 9 | INTx10 | R/W | 0h | Flag for Interrupt 8.10<br>Reset type: SYSRSn |
| 8 | INTx9 | R/W | 0h | Flag for Interrupt 8.9<br>Reset type: SYSRSn |
| 7 | INTx8 | R/W | 0h | Flag for Interrupt 8.8<br>Reset type: SYSRSn |
| 6 | INTx7 | R/W | 0h | Flag for Interrupt 8.7<br>Reset type: SYSRSn |
| 5 | INTx6 | R/W | 0h | Flag for Interrupt 8.6<br>Reset type: SYSRSn |
| 4 | INTx5 | R/W | 0h | Flag for Interrupt 8.5<br>Reset type: SYSRSn |
| 3 | INTx4 | R/W | 0h | Flag for Interrupt 8.4<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-47. PIEIFR8 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 2 | INTx3 | R/W | 0h | Flag for Interrupt 8.3<br>Reset type: SYSRSn |
| 1 | INTx2 | R/W | 0h | Flag for Interrupt 8.2<br>Reset type: SYSRSn |
| 0 | INTx1 | R/W | 0h | Flag for Interrupt 8.1<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.3.19 PIEIER9 Register (Offset = 12h) [Reset = 0000h]

PIEIER9 is shown in Figure 3-49 and described in Table 3-48.

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 3-49. PIEIER9 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| INTx16 | INTx15 | INTx14 | INTx13 | INTx12 | INTx11 | INTx10 | INTx9 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTx8 | INTx7 | INTx6 | INTx5 | INTx4 | INTx3 | INTx2 | INTx1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-48. PIEIER9 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | INTx16 | R/W | 0h | Enable for Interrupt 9.16<br>Reset type: SYSRSn |
| 14 | INTx15 | R/W | 0h | Enable for Interrupt 9.15<br>Reset type: SYSRSn |
| 13 | INTx14 | R/W | 0h | Enable for Interrupt 9.14<br>Reset type: SYSRSn |
| 12 | INTx13 | R/W | 0h | Enable for Interrupt 9.13<br>Reset type: SYSRSn |
| 11 | INTx12 | R/W | 0h | Enable for Interrupt 9.12<br>Reset type: SYSRSn |
| 10 | INTx11 | R/W | 0h | Enable for Interrupt 9.11<br>Reset type: SYSRSn |
| 9 | INTx10 | R/W | 0h | Enable for Interrupt 9.10<br>Reset type: SYSRSn |
| 8 | INTx9 | R/W | 0h | Enable for Interrupt 9.9<br>Reset type: SYSRSn |
| 7 | INTx8 | R/W | 0h | Enable for Interrupt 9.8<br>Reset type: SYSRSn |
| 6 | INTx7 | R/W | 0h | Enable for Interrupt 9.7<br>Reset type: SYSRSn |
| 5 | INTx6 | R/W | 0h | Enable for Interrupt 9.6<br>Reset type: SYSRSn |
| 4 | INTx5 | R/W | 0h | Enable for Interrupt 9.5<br>Reset type: SYSRSn |
| 3 | INTx4 | R/W | 0h | Enable for Interrupt 9.4<br>Reset type: SYSRSn |
| 2 | INTx3 | R/W | 0h | Enable for Interrupt 9.3<br>Reset type: SYSRSn |
| 1 | INTx2 | R/W | 0h | Enable for Interrupt 9.2<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-48. PIEIER9 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | INTx1 | R/W | 0h | Enable for Interrupt 9.1<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.3.20 PIEIFR9 Register (Offset = 13h) [Reset = 0000h]

PIEIFR9 is shown in Figure 3-50 and described in Table 3-49.

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 3-50. PIEIFR9 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| INTx16 | INTx15 | INTx14 | INTx13 | INTx12 | INTx11 | INTx10 | INTx9 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTx8 | INTx7 | INTx6 | INTx5 | INTx4 | INTx3 | INTx2 | INTx1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-49. PIEIFR9 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | INTx16 | R/W | 0h | Flag for Interrupt 9.16<br>Reset type: SYSRSn |
| 14 | INTx15 | R/W | 0h | Flag for Interrupt 9.15<br>Reset type: SYSRSn |
| 13 | INTx14 | R/W | 0h | Flag for Interrupt 9.14<br>Reset type: SYSRSn |
| 12 | INTx13 | R/W | 0h | Flag for Interrupt 9.13<br>Reset type: SYSRSn |
| 11 | INTx12 | R/W | 0h | Flag for Interrupt 9.12<br>Reset type: SYSRSn |
| 10 | INTx11 | R/W | 0h | Flag for Interrupt 9.11<br>Reset type: SYSRSn |
| 9 | INTx10 | R/W | 0h | Flag for Interrupt 9.10<br>Reset type: SYSRSn |
| 8 | INTx9 | R/W | 0h | Flag for Interrupt 9.9<br>Reset type: SYSRSn |
| 7 | INTx8 | R/W | 0h | Flag for Interrupt 9.8<br>Reset type: SYSRSn |
| 6 | INTx7 | R/W | 0h | Flag for Interrupt 9.7<br>Reset type: SYSRSn |
| 5 | INTx6 | R/W | 0h | Flag for Interrupt 9.6<br>Reset type: SYSRSn |
| 4 | INTx5 | R/W | 0h | Flag for Interrupt 9.5<br>Reset type: SYSRSn |
| 3 | INTx4 | R/W | 0h | Flag for Interrupt 9.4<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-49. PIEIFR9 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 2 | INTx3 | R/W | 0h | Flag for Interrupt 9.3<br>Reset type: SYSRSn |
| 1 | INTx2 | R/W | 0h | Flag for Interrupt 9.2<br>Reset type: SYSRSn |
| 0 | INTx1 | R/W | 0h | Flag for Interrupt 9.1<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.3.21 PIEIER10 Register (Offset = 14h) [Reset = 0000h]

PIEIER10 is shown in Figure 3-51 and described in Table 3-50.

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 3-51. PIEIER10 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| INTx16 | INTx15 | INTx14 | INTx13 | INTx12 | INTx11 | INTx10 | INTx9 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTx8 | INTx7 | INTx6 | INTx5 | INTx4 | INTx3 | INTx2 | INTx1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-50. PIEIER10 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | INTx16 | R/W | 0h | Enable for Interrupt 10.16<br>Reset type: SYSRSn |
| 14 | INTx15 | R/W | 0h | Enable for Interrupt 10.15<br>Reset type: SYSRSn |
| 13 | INTx14 | R/W | 0h | Enable for Interrupt 10.14<br>Reset type: SYSRSn |
| 12 | INTx13 | R/W | 0h | Enable for Interrupt 10.13<br>Reset type: SYSRSn |
| 11 | INTx12 | R/W | 0h | Enable for Interrupt 10.12<br>Reset type: SYSRSn |
| 10 | INTx11 | R/W | 0h | Enable for Interrupt 10.11<br>Reset type: SYSRSn |
| 9 | INTx10 | R/W | 0h | Enable for Interrupt 10.10<br>Reset type: SYSRSn |
| 8 | INTx9 | R/W | 0h | Enable for Interrupt 10.9<br>Reset type: SYSRSn |
| 7 | INTx8 | R/W | 0h | Enable for Interrupt 10.8<br>Reset type: SYSRSn |
| 6 | INTx7 | R/W | 0h | Enable for Interrupt 10.7<br>Reset type: SYSRSn |
| 5 | INTx6 | R/W | 0h | Enable for Interrupt 10.6<br>Reset type: SYSRSn |
| 4 | INTx5 | R/W | 0h | Enable for Interrupt 10.5<br>Reset type: SYSRSn |
| 3 | INTx4 | R/W | 0h | Enable for Interrupt 10.4<br>Reset type: SYSRSn |
| 2 | INTx3 | R/W | 0h | Enable for Interrupt 10.3<br>Reset type: SYSRSn |
| 1 | INTx2 | R/W | 0h | Enable for Interrupt 10.2<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-50. PIEIER10 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | INTx1 | R/W | 0h | Enable for Interrupt 10.1<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.3.22 PIEIFR10 Register (Offset = 15h) [Reset = 0000h]

PIEIFR10 is shown in Figure 3-52 and described in Table 3-51.

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 3-52. PIEIFR10 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| INTx16 | INTx15 | INTx14 | INTx13 | INTx12 | INTx11 | INTx10 | INTx9 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTx8 | INTx7 | INTx6 | INTx5 | INTx4 | INTx3 | INTx2 | INTx1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-51. PIEIFR10 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | INTx16 | R/W | 0h | Flag for Interrupt 10.16<br>Reset type: SYSRSn |
| 14 | INTx15 | R/W | 0h | Flag for Interrupt 10.15<br>Reset type: SYSRSn |
| 13 | INTx14 | R/W | 0h | Flag for Interrupt 10.14<br>Reset type: SYSRSn |
| 12 | INTx13 | R/W | 0h | Flag for Interrupt 10.13<br>Reset type: SYSRSn |
| 11 | INTx12 | R/W | 0h | Flag for Interrupt 10.12<br>Reset type: SYSRSn |
| 10 | INTx11 | R/W | 0h | Flag for Interrupt 10.11<br>Reset type: SYSRSn |
| 9 | INTx10 | R/W | 0h | Flag for Interrupt 10.10<br>Reset type: SYSRSn |
| 8 | INTx9 | R/W | 0h | Flag for Interrupt 10.9<br>Reset type: SYSRSn |
| 7 | INTx8 | R/W | 0h | Flag for Interrupt 10.8<br>Reset type: SYSRSn |
| 6 | INTx7 | R/W | 0h | Flag for Interrupt 10.7<br>Reset type: SYSRSn |
| 5 | INTx6 | R/W | 0h | Flag for Interrupt 10.6<br>Reset type: SYSRSn |
| 4 | INTx5 | R/W | 0h | Flag for Interrupt 10.5<br>Reset type: SYSRSn |
| 3 | INTx4 | R/W | 0h | Flag for Interrupt 10.4<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-51. PIEIFR10 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 2 | INTx3 | R/W | 0h | Flag for Interrupt 10.3<br>Reset type: SYSRSn |
| 1 | INTx2 | R/W | 0h | Flag for Interrupt 10.2<br>Reset type: SYSRSn |
| 0 | INTx1 | R/W | 0h | Flag for Interrupt 10.1<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.3.23 PIEIER11 Register (Offset = 16h) [Reset = 0000h]

PIEIER11 is shown in Figure 3-53 and described in Table 3-52.

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 3-53. PIEIER11 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| INTx16 | INTx15 | INTx14 | INTx13 | INTx12 | INTx11 | INTx10 | INTx9 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTx8 | INTx7 | INTx6 | INTx5 | INTx4 | INTx3 | INTx2 | INTx1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-52. PIEIER11 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | INTx16 | R/W | 0h | Enable for Interrupt 11.16<br>Reset type: SYSRSn |
| 14 | INTx15 | R/W | 0h | Enable for Interrupt 11.15<br>Reset type: SYSRSn |
| 13 | INTx14 | R/W | 0h | Enable for Interrupt 11.14<br>Reset type: SYSRSn |
| 12 | INTx13 | R/W | 0h | Enable for Interrupt 11.13<br>Reset type: SYSRSn |
| 11 | INTx12 | R/W | 0h | Enable for Interrupt 11.12<br>Reset type: SYSRSn |
| 10 | INTx11 | R/W | 0h | Enable for Interrupt 11.11<br>Reset type: SYSRSn |
| 9 | INTx10 | R/W | 0h | Enable for Interrupt 11.10<br>Reset type: SYSRSn |
| 8 | INTx9 | R/W | 0h | Enable for Interrupt 11.9<br>Reset type: SYSRSn |
| 7 | INTx8 | R/W | 0h | Enable for Interrupt 11.8<br>Reset type: SYSRSn |
| 6 | INTx7 | R/W | 0h | Enable for Interrupt 11.7<br>Reset type: SYSRSn |
| 5 | INTx6 | R/W | 0h | Enable for Interrupt 11.6<br>Reset type: SYSRSn |
| 4 | INTx5 | R/W | 0h | Enable for Interrupt 11.5<br>Reset type: SYSRSn |
| 3 | INTx4 | R/W | 0h | Enable for Interrupt 11.4<br>Reset type: SYSRSn |
| 2 | INTx3 | R/W | 0h | Enable for Interrupt 11.3<br>Reset type: SYSRSn |
| 1 | INTx2 | R/W | 0h | Enable for Interrupt 11.2<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-52. PIEIER11 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | INTx1 | R/W | 0h | Enable for Interrupt 11.1<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.3.24 PIEIFR11 Register (Offset = 17h) [Reset = 0000h]

PIEIFR11 is shown in Figure 3-54 and described in Table 3-53.

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 3-54. PIEIFR11 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| INTx16 | INTx15 | INTx14 | INTx13 | INTx12 | INTx11 | INTx10 | INTx9 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTx8 | INTx7 | INTx6 | INTx5 | INTx4 | INTx3 | INTx2 | INTx1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-53. PIEIFR11 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | INTx16 | R/W | 0h | Flag for Interrupt 11.16<br>Reset type: SYSRSn |
| 14 | INTx15 | R/W | 0h | Flag for Interrupt 11.15<br>Reset type: SYSRSn |
| 13 | INTx14 | R/W | 0h | Flag for Interrupt 11.14<br>Reset type: SYSRSn |
| 12 | INTx13 | R/W | 0h | Flag for Interrupt 11.13<br>Reset type: SYSRSn |
| 11 | INTx12 | R/W | 0h | Flag for Interrupt 11.12<br>Reset type: SYSRSn |
| 10 | INTx11 | R/W | 0h | Flag for Interrupt 11.11<br>Reset type: SYSRSn |
| 9 | INTx10 | R/W | 0h | Flag for Interrupt 11.10<br>Reset type: SYSRSn |
| 8 | INTx9 | R/W | 0h | Flag for Interrupt 11.9<br>Reset type: SYSRSn |
| 7 | INTx8 | R/W | 0h | Flag for Interrupt 11.8<br>Reset type: SYSRSn |
| 6 | INTx7 | R/W | 0h | Flag for Interrupt 11.7<br>Reset type: SYSRSn |
| 5 | INTx6 | R/W | 0h | Flag for Interrupt 11.6<br>Reset type: SYSRSn |
| 4 | INTx5 | R/W | 0h | Flag for Interrupt 11.5<br>Reset type: SYSRSn |
| 3 | INTx4 | R/W | 0h | Flag for Interrupt 11.4<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-53. PIEIFR11 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 2 | INTx3 | R/W | 0h | Flag for Interrupt 11.3<br>Reset type: SYSRSn |
| 1 | INTx2 | R/W | 0h | Flag for Interrupt 11.2<br>Reset type: SYSRSn |
| 0 | INTx1 | R/W | 0h | Flag for Interrupt 11.1<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.3.25 PIEIER12 Register (Offset = 18h) [Reset = 0000h]

PIEIER12 is shown in Figure 3-55 and described in Table 3-54.

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 3-55. PIEIER12 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| INTx16 | INTx15 | INTx14 | INTx13 | INTx12 | INTx11 | INTx10 | INTx9 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTx8 | INTx7 | INTx6 | INTx5 | INTx4 | INTx3 | INTx2 | INTx1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-54. PIEIER12 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | INTx16 | R/W | 0h | Enable for Interrupt 12.16<br>Reset type: SYSRSn |
| 14 | INTx15 | R/W | 0h | Enable for Interrupt 12.15<br>Reset type: SYSRSn |
| 13 | INTx14 | R/W | 0h | Enable for Interrupt 12.14<br>Reset type: SYSRSn |
| 12 | INTx13 | R/W | 0h | Enable for Interrupt 12.13<br>Reset type: SYSRSn |
| 11 | INTx12 | R/W | 0h | Enable for Interrupt 12.12<br>Reset type: SYSRSn |
| 10 | INTx11 | R/W | 0h | Enable for Interrupt 12.11<br>Reset type: SYSRSn |
| 9 | INTx10 | R/W | 0h | Enable for Interrupt 12.10<br>Reset type: SYSRSn |
| 8 | INTx9 | R/W | 0h | Enable for Interrupt 12.9<br>Reset type: SYSRSn |
| 7 | INTx8 | R/W | 0h | Enable for Interrupt 12.8<br>Reset type: SYSRSn |
| 6 | INTx7 | R/W | 0h | Enable for Interrupt 12.7<br>Reset type: SYSRSn |
| 5 | INTx6 | R/W | 0h | Enable for Interrupt 12.6<br>Reset type: SYSRSn |
| 4 | INTx5 | R/W | 0h | Enable for Interrupt 12.5<br>Reset type: SYSRSn |
| 3 | INTx4 | R/W | 0h | Enable for Interrupt 12.4<br>Reset type: SYSRSn |
| 2 | INTx3 | R/W | 0h | Enable for Interrupt 12.3<br>Reset type: SYSRSn |
| 1 | INTx2 | R/W | 0h | Enable for Interrupt 12.2<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-54. PIEIER12 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | INTx1 | R/W | 0h | Enable for Interrupt 12.1<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.3.26 PIEIFR12 Register (Offset = 19h) [Reset = 0000h]

PIEIFR12 is shown in Figure 3-56 and described in Table 3-55.

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 3-56. PIEIFR12 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| INTx16 | INTx15 | INTx14 | INTx13 | INTx12 | INTx11 | INTx10 | INTx9 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTx8 | INTx7 | INTx6 | INTx5 | INTx4 | INTx3 | INTx2 | INTx1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-55. PIEIFR12 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | INTx16 | R/W | 0h | Flag for Interrupt 12.16<br>Reset type: SYSRSn |
| 14 | INTx15 | R/W | 0h | Flag for Interrupt 12.15<br>Reset type: SYSRSn |
| 13 | INTx14 | R/W | 0h | Flag for Interrupt 12.14<br>Reset type: SYSRSn |
| 12 | INTx13 | R/W | 0h | Flag for Interrupt 12.13<br>Reset type: SYSRSn |
| 11 | INTx12 | R/W | 0h | Flag for Interrupt 12.12<br>Reset type: SYSRSn |
| 10 | INTx11 | R/W | 0h | Flag for Interrupt 12.11<br>Reset type: SYSRSn |
| 9 | INTx10 | R/W | 0h | Flag for Interrupt 12.10<br>Reset type: SYSRSn |
| 8 | INTx9 | R/W | 0h | Flag for Interrupt 12.9<br>Reset type: SYSRSn |
| 7 | INTx8 | R/W | 0h | Flag for Interrupt 12.8<br>Reset type: SYSRSn |
| 6 | INTx7 | R/W | 0h | Flag for Interrupt 12.7<br>Reset type: SYSRSn |
| 5 | INTx6 | R/W | 0h | Flag for Interrupt 12.6<br>Reset type: SYSRSn |
| 4 | INTx5 | R/W | 0h | Flag for Interrupt 12.5<br>Reset type: SYSRSn |
| 3 | INTx4 | R/W | 0h | Flag for Interrupt 12.4<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-55. PIEIFR12 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 2 | INTx3 | R/W | 0h | Flag for Interrupt 12.3<br>Reset type: SYSRSn |
| 1 | INTx2 | R/W | 0h | Flag for Interrupt 12.2<br>Reset type: SYSRSn |
| 0 | INTx1 | R/W | 0h | Flag for Interrupt 12.1<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.4 WD_REGS Registers

Table 3-56 lists the memory-mapped registers for the WD_REGS registers. All register offset addresses not listed in Table 3-56 should be considered as reserved locations and the register contents should not be modified.

**Table 3-56. WD_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 22h | SCSR | System Control & Status Register | EALLOW | Go |
| 23h | WDCNTR | Watchdog Counter Register | EALLOW | Go |
| 25h | WDKEY | Watchdog Reset Key Register | EALLOW | Go |
| 29h | WDCR | Watchdog Control Register<br>Note: IORSn reset type is asserted when XRSn or CPU1.SYSRSn or CPU1.WDRSn or CPU1.NMIRSn is asserted. | EALLOW | Go |
| 2Ah | WDWCR | Watchdog Windowed Control Register | EALLOW | Go |

Complex bit access types are encoded to fit into small table cells. Table 3-57 shows the codes that are used for access types in this section.

**Table 3-57. WD_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| R-0 | R<br>-0 | Read<br>Returns 0s |
| Write Type | | |
| W | W | Write |
| W1C | W<br>1C | Write<br>1 to clear |
| W1S | W<br>1S | Write<br>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. |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.4.1 SCSR Register (Offset = 22h) [Reset = 0005h]

SCSR is shown in Figure 3-57 and described in Table 3-58.

Return to the Summary Table.

It is recommended to only use 16 bit accesses to write to this register. Use a read-modify-write instruction may inadvertently clear other bits.

**Figure 3-57. 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/W1C-1h |

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

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-3 | RESERVED | R-0 | 0h | Reserved |
| 2 | WDINTS | R | 1h | Watchdog interrupt (WDINTn) status signal. This is a read only bit reflecting the current state of the WDINTn signal from the watchdog block (after synchronization with SYSCLKOUT). If this bit is 1, the watchdog interrupt is not active. If this bit is 0, then the watchdog interrupt is active.<br>Note: ,If the WDINTn signal is used to wake up from IDLE or STANDBY condition, then the user should make sure that the WDINTn signal goes back high again before attempting to go back into IDLE or STANDBY mode. Reading this bit will tell the user the current state of this signal.<br>Reset type: SYSRSn |
| 1 | WDENINT | R/W | 0h | If this bit is set to 1, the watchdog reset (WDRSTn) output signal is disabled and the watchdog interrupt (WDINTn) output signal is enabled. If this bit is zero, then the WDRSTn output signal is enabled and the WDINTn output signal is disabled. This is the default state on system reset (SYSRSn).<br>Reset type: SYSRSn |
| 0 | WDOVERRIDE | R/W1C | 1h | If this bit is set to 1, the user is allowed to change the state of the Watchdog disable (WDDIS) bit in the Watchdog Control (WDCR) register. If the WDOVERRIDE bit is cleared, by writing a 1 the WDDIS bit cannot be modified by the user. Writing a 0 will have no effect. If this bit is cleared, then it will remain in this state until a reset occurs. The current state of this bit is readable by the user.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.4.2 WDCNTR Register (Offset = 23h) [Reset = 0000h]

WDCNTR is shown in Figure 3-58 and described in Table 3-59.

Return to the Summary Table.

Watchdog Counter Register

**Figure 3-58. 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 3-59. WDCNTR Register Field Descriptions**

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



Copyright © 2024 Texas Instruments Incorporated

### 3.17.4.3 WDKEY Register (Offset = 25h) [Reset = 0000h]

WDKEY is shown in Figure 3-59 and described in Table 3-60.

Return to the Summary Table.

Watchdog Reset Key Register

**Figure 3-59. 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 3-60. WDKEY Register Field Descriptions**

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



Copyright © 2024 Texas Instruments Incorporated

### 3.17.4.4 WDCR Register (Offset = 29h) [Reset = 0000h]

WDCR is shown in Figure 3-60 and described in Table 3-61.

Return to the Summary Table.

Watchdog Control Register
This memory mapped register requires a delay of 69 SYSCLK cycles between subsequent writes to the register, otherwise a second write can be lost. This delay can be realized by adding 69 NOP instructions.

**Figure 3-60. WDCR Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | WDDIS | WDCHK | | | WDPS | | |
| R/W1S-0h | R/W-0h | R-0/W-0h | | | R/W-0h | | |

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

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R-0 | 0h | Reserved |
| 7 | RESERVED | R/W1S | 0h | Reserved |
| 6 | WDDIS | R/W | 0h | Writing a 1 to this bit will disable the watchdog module. Writing a 0 will enable the module. This bit can only be modified if the WDOVERRIDE bit in the SCSR2 register is set to 1. On reset, the watchdog module is enabled.<br>Reset type: IORSn |
| 5-3 | WDCHK | R-0/W | 0h | The user must ALWAYS write 1,0,1 to these bits whenever a write to this register is performed. Writing any other value will cause an immediate reset to the core (if WD enabled).<br>Reset type: IORSn |
| 2-0 | WDPS | R/W | 0h | These bits configure the watchdog counter clock (WDCLK) rate relative to INTOSC1/512:<br>000 WDCLK = INTOSC1/512/1<br>001 WDCLK = INTOSC1/512/1<br>010 WDCLK = INTOSC1/512/2<br>011 WDCLK = INTOSC1/512/4<br>100 WDCLK = INTOSC1/512/8<br>101 WDCLK = INTOSC1/512/16<br>110 WDCLK = INTOSC1/512/32<br>111 WDCLK = INTOSC1/512/64<br>Reset type: IORSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.4.5 WDWCR Register (Offset = 2Ah) [Reset = 0000h]

WDWCR is shown in Figure 3-61 and described in Table 3-62.

Return to the Summary Table.

Watchdog Windowed Control Register

**Figure 3-61. WDWCR Register**

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

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

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-9 | RESERVED | R-0 | 0h | Reserved |
| 8 | FIRSTKEY | R | 0h | This bit indicates if the 1st valid WDKEY (0x55 + 0xAA) got detected after MIN was configured to a non-zero value<br>0: First Valid Key after non-zero MIN configuration has not happened yet<br>1: First Valid key after non-zero MIN configuration got detected<br>Notes:<br>[1] If MIN = 0, this bit is never set<br>[2] If MIN is changed back to 0x0 from a non-zero value, this bit is auto-cleared<br>[3] This bit is added for debug purposes only<br>Reset type: IORSn |
| 7-0 | MIN | R/W | 0h | Watchdog Window Threshold<br>These bits specify the lower limit of the watchdog counter reset window. If the counter reset via the WDKEY register before the counter value reaches the value in this register, the watchdog immediately triggers a reset or interrupt.<br>Reset type: IORSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.5 NMI_INTRUPT_REGS Registers

Table 3-63 lists the memory-mapped registers for the NMI_INTRUPT_REGS registers. All register offset addresses not listed in Table 3-63 should be considered as reserved locations and the register contents should not be modified.

**Table 3-63. NMI_INTRUPT_REGS Registers**

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

Complex bit access types are encoded to fit into small table cells. Table 3-64 shows the codes that are used for access types in this section.

**Table 3-64. NMI_INTRUPT_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| R-0 | R<br>-0 | Read<br>Returns 0s |
| Write Type | | |
| W | W | Write |
| W1S | W<br>1S | Write<br>1 to set |
| Reset or Default Value | | |
| *-n* | | Value after reset or the default value |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.5.1 NMICFG Register (Offset = 0h) [Reset = 0000h]

NMICFG is shown in Figure 3-62 and described in Table 3-65.

Return to the Summary Table.

NMI Configuration Register

**Figure 3-62. 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 3-65. 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.<br>0 NMI disabled<br>1 NMI enabled<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.5.2 NMIFLG Register (Offset = 1h) [Reset = 0000h]

NMIFLG is shown in Figure 3-63 and described in Table 3-66.

Return to the Summary Table.

NMI Flag Register (XRSn Clear)

**Figure 3-63. NMIFLG Register**

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

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

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | RESERVED | R-0 | 0h | Reserved |
| 11 | RESERVED | R | 0h | Reserved |
| 10 | CPU2NMIWDRSn | R | 0h | CPU2 NMIWDRSn Reset Indication Flag: This bits indicates if CPU2s NMIWDRSn was fired or not.<br>0 No CPU2.NMIWDRsn was fired<br>1 CPU2.NMIWDRSn was fired to CPU2<br>Note:<br>[1] This bits is reserved for CPU2.NMIFLG register<br>Reset type: XRSn |
| 9 | CPU2WDRSn | R | 0h | CPU2 WDRSn Reset Indication Flag: This bits indicates if CPU2s WDRSn was fired or not.<br>0 No CPU2.WDRsn was fired<br>1 CPU2.WDRSn was fired to CPU2<br>Note:<br>[1] This bits is reserved for CPU2.NMIFLG register<br>Reset type: XRSn |
| 8 | CLBNMI | R | 0h | Configurable Logic Block NMI Flag: This bit indicates if an NMI was generated by the Configurable Logic Block. This bit can only be cleared by the user writing to the corresponding clear bit in the NMIFLGCLR register or by an XRSn reset:<br>0,No Configurable Logic Block NMI pending<br>1,Configurable Logic Block NMI generated<br>Reset type: XRSn |
| 7 | RESERVED | R | 0h | Reserved |
| 6 | PIEVECTERR | R | 0h | PIE Vector Fetch Error Flag: This bit indicates if an error occurred on an Vector Fect by the other CPU in the device. For example, CPU1.NMIWD gets an NMI on an Vector fetch Error on CPU2. This bit can only be cleared by the user writing to the corresponding clear bit in the NMIFLGCLR register or by an XRSn reset:<br>0,No Vector Fetch Error condition (on the other CPU) pending<br>1,Vector Fetch error condition (on the other CPU) generated<br>Reset type: XRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-66. NMIFLG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 5 | CPU2HWBISTERR | R | 0h | HW BIST Error NMI Flag: This bit indicates if the time out error or a signature mismatch error condition during hardware BIST of C28 CPU2 occurred. This bit can only be cleared by the user writing to the corresponding clear bit in the NMIFLGCLR register or by an XRSn reset:<br>0,No C28 HWBIST error condition pending<br>1,C28 BIST error condition generated<br>Reset type: XRSn |
| 4 | CPU1HWBISTERR | R | 0h | HW BIST Error NMI Flag: This bit indicates if the time out error or a signature mismatch error condition during hardware BIST of C28 CPU1 occurred. This bit can only be cleared by the user writing to the corresponding clear bit in the NMIFLGCLR register or by an XRSn reset:<br>0,No C28 HWBIST error condition pending<br>1,C28 BIST error condition generated<br>Reset type: XRSn |
| 3 | FLUNCERR | R | 0h | Flash Uncorrectable Error NMI Flag: This bit indicates if an uncorrectable error occurred on a C28 Flash access and that condition is latched. This bit can only be cleared by the user writing to the corresponding clear bit in the NMIFLGCLR register or by an XRSn reset:<br>0,No C28 Flash uncorrectable error condition pending<br>1,C28 Flash uncorrectable error condition generated<br>Reset type: XRSn |
| 2 | RAMUNCERR | R | 0h | RAM Uncorrectable Error NMI Flag: This bit indicates if an uncorrectable error occurred on a RAM access (by any master) and that condition is latched. This bit can only be cleared by the user writing to the corresponding clear bit in the NMIFLGCLR register or by an XRSn reset:<br>0,No RAM uncorrectable error condition pending<br>1,RAM uncorrectable error condition generated<br>Reset type: XRSn |
| 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 an XRSn reset:<br>0,No CLOCKFAIL Condition Pending<br>1,CLOCKFAIL Condition Generated<br>Reset type: XRSn |
| 0 | NMIINT | R | 0h | NMI Interrupt Flag: This bit indicates if an NMI interrupt was generated. This bit can only be cleared by the user writing to the respective bit in the NMIFLGCLR register or by an XRSn reset:<br>0 No NMI Interrupt Generated<br>1 NMI Interrupt Generated<br>No further NMI interrupts pulses are generated until this flag is cleared by the user.<br>Reset type: XRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.5.3 NMIFLGCLR Register (Offset = 2h) [Reset = 0000h]

NMIFLGCLR is shown in Figure 3-64 and described in Table 3-67.

Return to the Summary Table.

NMI Flag Clear Register

**Figure 3-64. NMIFLGCLR Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | RESERVED | CPU2NMIWDRSn | CPU2WDRSn | CLBNMI |
| R-0-0h | | | | 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 | PIEVECTERR | CPU2HWBISTERR | CPU1HWBISTERR | FLUNCERR | RAMUNCERR | CLOCKFAIL | NMIINT |
| 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 3-67. NMIFLGCLR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | RESERVED | R-0 | 0h | Reserved |
| 11 | RESERVED | R-0/W1S | 0h | Reserved |
| 10 | CPU2NMIWDRSn | 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.<br>Notes:<br>[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.<br>[2] Users should clear the pending FAIL flag first and then clear the NMIINT flag.<br>Reset type: SYSRSn |
| 9 | CPU2WDRSn | 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.<br>Notes:<br>[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.<br>[2] Users should clear the pending FAIL flag first and then clear the NMIINT flag.<br>[3] CPU2WDRSn and CPU2NMIWDRSn bits are reserved for CPU2.NMIFLGCLR registers<br>Reset type: SYSRSn |
| 8 | CLBNMI | 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.<br>Notes:<br>[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.<br>[2] Users should clear the pending FAIL flag first and then clear the NMIINT flag.<br>Reset type: SYSRSn |
| 7 | RESERVED | R-0/W1S | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

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

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 6 | PIEVECTERR | 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.<br>Notes:<br>[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.<br>[2] Users should clear the pending FAIL flag first and then clear the NMIINT flag.<br>Reset type: SYSRSn |
| 5 | CPU2HWBISTERR | 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.<br>Notes:<br>[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.<br>[2] Users should clear the pending FAIL flag first and then clear the NMIINT flag.<br>Reset type: SYSRSn |
| 4 | CPU1HWBISTERR | 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.<br>Notes:<br>[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.<br>[2] Users should clear the pending FAIL flag first and then clear the NMIINT flag.<br>Reset type: SYSRSn |
| 3 | FLUNCERR | 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.<br>Notes:<br>[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.<br>[2] Users should clear the pending FAIL flag first and then clear the NMIINT flag.<br>Reset type: SYSRSn |
| 2 | RAMUNCERR | 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.<br>Notes:<br>[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.<br>[2] Users should clear the pending FAIL flag first and then clear the NMIINT flag.<br>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.<br>Notes:<br>[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.<br>[2] Users should clear the pending FAIL flag first and then clear the NMIINT flag.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-67. 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.<br>Notes:<br>[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.<br>[2] Users should clear the pending FAIL flag first and then clear the NMIINT flag.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.5.4 NMIFLGFRC Register (Offset = 3h) [Reset = 0000h]

NMIFLGFRC is shown in Figure 3-65 and described in Table 3-68.

Return to the Summary Table.

NMI Flag Force Register

**Figure 3-65. NMIFLGFRC Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | RESERVED | CPU2NMIWDRSn | CPU2WDRSn | CLBNMI |
| R-0-0h | | | | 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 | PIEVECTERR | CPU2HWBISTERR | CPU1HWBISTERR | FLUNCERR | RAMUNCERR | CLOCKFAIL | 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-0-0h |

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

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | RESERVED | R-0 | 0h | Reserved |
| 11 | RESERVED | R-0/W1S | 0h | Reserved |
| 10 | CPU2NMIWDRSn | 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.<br>Note:<br>[1] CPU2WDRSn and CPU2NMIWDRSn bits are reserved for CPU2.NMIFLGCLR registers<br>Reset type: SYSRSn |
| 9 | CPU2WDRSn | 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.<br>Note:<br>[1] CPU2WDRSn and CPU2NMIWDRSn bits are reserved for CPU2.NMIFLGCLR registers<br>Reset type: SYSRSn |
| 8 | CLBNMI | 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.<br>Reset type: SYSRSn |
| 7 | RESERVED | R-0/W1S | 0h | Reserved |
| 6 | PIEVECTERR | 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.<br>Reset type: SYSRSn |
| 5 | CPU2HWBISTERR | 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.<br>Reset type: SYSRSn |
| 4 | CPU1HWBISTERR | 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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

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

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3 | FLUNCERR | 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.<br>Reset type: SYSRSn |
| 2 | RAMUNCERR | 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.<br>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.<br>Reset type: SYSRSn |
| 0 | RESERVED | R-0 | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.5.5 NMIWDCNT Register (Offset = 4h) [Reset = 0000h]

NMIWDCNT is shown in Figure 3-66 and described in Table 3-69.

Return to the Summary Table.

NMI Watchdog Counter Register

**Figure 3-66. 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 3-69. NMIWDCNT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | NMIWDCNT | R | 0h | 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.<br>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.<br>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.<br>The counter is clocked at the SYSCLKOUT rate.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.5.6 NMIWDPRD Register (Offset = 5h) [Reset = FFFFh]

NMIWDPRD is shown in Figure 3-67 and described in Table 3-70.

Return to the Summary Table.

NMI Watchdog Period Register

**Figure 3-67. 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 3-70. NMIWDPRD Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | NMIWDPRD | R/W | FFFFh | 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.<br>Writing a PERIOD value that is smaller then the current counter value will automatically force an NMIRSn and hence reset the watchdog counter.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.5.7 NMISHDFLG Register (Offset = 6h) [Reset = 0000h]

NMISHDFLG is shown in Figure 3-68 and described in Table 3-71.

Return to the Summary Table.

NMI Shadow Flag Register

**Figure 3-68. NMISHDFLG Register**

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

**Table 3-71. NMISHDFLG Register Field Descriptions**

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



Copyright © 2024 Texas Instruments Incorporated

**Table 3-71. NMISHDFLG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 6 | PIEVECTERR | R | 0h | Shadow NMI Flags: When an NMIFLG bit is set due to any of the possible NMI source in the device, the corresponding bit in this register is also set. Note that NMIFLGFRC and NMIFLGCLR register also affects the bits of this register in the same way as they do for the NMIFLG register. This register is only reset by a PORESETn reset.<br>Notes:<br>[1] This register is added to keep the definition of System Control Reset Cause Register Clean.<br>Reset type: PORESETn |
| 5 | CPU2HWBISTERR | R | 0h | Shadow NMI Flags: When an NMIFLG bit is set due to any of the possible NMI source in the device, the corresponding bit in this register is also set. Note that 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 only reset by a PORESETn reset.<br>Notes:<br>[1] This register is added to keep the definition of System Control Reset Cause Register Clean.<br>Reset type: PORESETn |
| 4 | CPU1HWBISTERR | R | 0h | Shadow NMI Flags: When an NMIFLG bit is set due to any of the possible NMI source in the device, the corresponding bit in this register is also set. Note that 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 only reset by a PORESETn reset.<br>Notes:<br>[1] This register is added to keep the definition of System Control Reset Cause Register Clean.<br>Reset type: PORESETn |
| 3 | FLUNCERR | R | 0h | Shadow NMI Flags: When an NMIFLG bit is set due to any of the possible NMI source in the device, the corresponding bit in this register is also set. Note that 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 only reset by a PORESETn reset.<br>Notes:<br>[1] This register is added to keep the definition of System Control Reset Cause Register Clean.<br>Reset type: PORESETn |
| 2 | RAMUNCERR | R | 0h | Shadow NMI Flags: When an NMIFLG bit is set due to any of the possible NMI source in the device, the corresponding bit in this register is also set. Note that 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 only reset by a PORESETn reset.<br>Notes:<br>[1] This register is added to keep the definition of System Control Reset Cause Register Clean.<br>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 only reset by a PORESETn reset.<br>Notes:<br>[1] This register is added to keep the definition of System Control Reset Cause Register Clean.<br>Reset type: PORESETn |
| 0 | RESERVED | R-0 | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.6 XINT_REGS Registers

Table 3-72 lists the memory-mapped registers for the XINT_REGS registers. All register offset addresses not listed in Table 3-72 should be considered as reserved locations and the register contents should not be modified.

**Table 3-72. XINT_REGS Registers**

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

Complex bit access types are encoded to fit into small table cells. Table 3-73 shows the codes that are used for access types in this section.

**Table 3-73. XINT_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| R-0 | R<br>-0 | Read<br>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. |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.6.1 XINT1CR Register (Offset = 0h) [Reset = 0000h]

XINT1CR is shown in Figure 3-69 and described in Table 3-74.

Return to the Summary Table.

XINT1 configuration register

**Figure 3-69. 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 3-74. 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<br>01: Interrupt is selected as positive edge triggered<br>10: Interrupt is selected as negative edge triggered<br>11: Interrupt is selected as positive or negative edge triggered<br>Reset type: SYSRSn |
| 1 | RESERVED | R-0 | 0h | Reserved |
| 0 | ENABLE | R/W | 0h | 0: Interrupt Disabled<br>1: Interrupt Enabled<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.6.2 XINT2CR Register (Offset = 1h) [Reset = 0000h]

XINT2CR is shown in Figure 3-70 and described in Table 3-75.

Return to the Summary Table.

XINT2 configuration register

**Figure 3-70. 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 3-75. 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<br>01: Interrupt is selected as positive edge triggered<br>10: Interrupt is selected as negative edge triggered<br>11: Interrupt is selected as positive or negative edge triggered<br>Reset type: SYSRSn |
| 1 | RESERVED | R-0 | 0h | Reserved |
| 0 | ENABLE | R/W | 0h | 0: Interrupt Disabled<br>1: Interrupt Enabled<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.6.3 XINT3CR Register (Offset = 2h) [Reset = 0000h]

XINT3CR is shown in Figure 3-71 and described in Table 3-76.

Return to the Summary Table.

XINT3 configuration register

**Figure 3-71. 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 3-76. 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<br>01: Interrupt is selected as positive edge triggered<br>10: Interrupt is selected as negative edge triggered<br>11: Interrupt is selected as positive or negative edge triggered<br>Reset type: SYSRSn |
| 1 | RESERVED | R-0 | 0h | Reserved |
| 0 | ENABLE | R/W | 0h | 0: Interrupt Disabled<br>1: Interrupt Enabled<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.6.4 XINT4CR Register (Offset = 3h) [Reset = 0000h]

XINT4CR is shown in Figure 3-72 and described in Table 3-77.

Return to the Summary Table.

XINT4 configuration register

**Figure 3-72. 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 3-77. 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<br>01: Interrupt is selected as positive edge triggered<br>10: Interrupt is selected as negative edge triggered<br>11: Interrupt is selected as positive or negative edge triggered<br>Reset type: SYSRSn |
| 1 | RESERVED | R-0 | 0h | Reserved |
| 0 | ENABLE | R/W | 0h | 0: Interrupt Disabled<br>1: Interrupt Enabled<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.6.5 XINT5CR Register (Offset = 4h) [Reset = 0000h]

XINT5CR is shown in Figure 3-73 and described in Table 3-78.

Return to the Summary Table.

XINT5 configuration register

**Figure 3-73. 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 3-78. 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<br>01: Interrupt is selected as positive edge triggered<br>10: Interrupt is selected as negative edge triggered<br>11: Interrupt is selected as positive or negative edge triggered<br>Reset type: SYSRSn |
| 1 | RESERVED | R-0 | 0h | Reserved |
| 0 | ENABLE | R/W | 0h | 0: Interrupt Disabled<br>1: Interrupt Enabled<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.6.6 XINT1CTR Register (Offset = 8h) [Reset = 0000h]

XINT1CTR is shown in Figure 3-74 and described in Table 3-79.

Return to the Summary Table.

XINT1 counter register

**Figure 3-74. 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 3-79. 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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.6.7 XINT2CTR Register (Offset = 9h) [Reset = 0000h]

XINT2CTR is shown in Figure 3-75 and described in Table 3-80.

Return to the Summary Table.

XINT2 counter register

**Figure 3-75. 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 3-80. 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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.6.8 XINT3CTR Register (Offset = Ah) [Reset = 0000h]

XINT3CTR is shown in Figure 3-76 and described in Table 3-81.

Return to the Summary Table.

XINT3 counter register

**Figure 3-76. 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 3-81. 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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.7 SYNC_SOC_REGS Registers

Table 3-82 lists the memory-mapped registers for the SYNC_SOC_REGS registers. All register offset addresses not listed in Table 3-82 should be considered as reserved locations and the register contents should not be modified.

**Table 3-82. SYNC_SOC_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | SYNCSELECT | Sync Input and Output Select Register | EALLOW | Go |
| 2h | ADCSOCOUTSELECT | External ADC (Off Chip) SOC Select Register | EALLOW | Go |
| 4h | SYNCSOCLOCK | SYNCSEL and EXTADCSOC Select Lock register | EALLOW | Go |

Complex bit access types are encoded to fit into small table cells. Table 3-83 shows the codes that are used for access types in this section.

**Table 3-83. SYNC_SOC_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| R-0 | R<br>-0 | Read<br>Returns 0s |
| Write Type | | |
| W | W | Write |
| WSonce | W<br>Sonce | Write<br>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. |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.7.1 SYNCSELECT Register (Offset = 0h) [Reset = 00000000h]

SYNCSELECT is shown in Figure 3-77 and described in Table 3-84.

Return to the Summary Table.

Sync Input and Output Select Register

**Figure 3-77. SYNCSELECT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | SYNCOUT | | RESERVED | | |
| R-0-0h | | | R/W-0h | | R-0-0h | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | ECAP4SYNCIN | | | ECAP1SYNCIN | | | EPWM10SYNCIN |
| R-0-0h | R/W-0h | | | R/W-0h | | | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EPWM10SYNCIN | | EPWM7SYNCIN | | | EPWM4SYNCIN | | |
| R/W-0h | | R/W-0h | | | R/W-0h | | |

**Table 3-84. SYNCSELECT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-29 | RESERVED | R-0 | 0h | Reserved |
| 28-27 | SYNCOUT | R/W | 0h | Select Syncout Source:<br>00: EPWM1SYNCOUT selected<br>01: EPWM4SYNCOUT selected<br>10: EPWM7SYNCOUT selected<br>11: EPWM10SYNCOUT selected<br>Reset type: CPU1.SYSRSn |
| 26-16 | RESERVED | R-0 | 0h | Reserved |
| 15 | RESERVED | R-0 | 0h | Reserved |
| 14-12 | ECAP4SYNCIN | R/W | 0h | Selects Sync Input Source for ECAP4:<br>000: EPWM1SYNCOUT selected<br>001: EPWM4SYNCOUT selected<br>010: EPWM7SYNCOUT selected<br>011: EPWM10SYNCOUT selected<br>100: ECAP1SYNCOUT selected<br>101: EXTSYNCIN1 selected<br>110: EXTSYNCIN2 selected<br>111: Reserved<br>Notes:<br>[1] Reserved position defaults to 000 selection<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-84. SYNCSELECT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 11-9 | ECAP1SYNCIN | R/W | 0h | Selects Sync Input Source for ECAP1:<br>000: EPWM1SYNCOUT selected<br>001: EPWM4SYNCOUT selected<br>010: EPWM7SYNCOUT selected<br>011: EPWM10SYNCOUT selected<br>100: ECAP1SYNCOUT selected (Reserved)<br>101: EXTSYNCIN1 selected<br>110: EXTSYNCIN2 selected<br>111: Reserved<br>Notes:<br>[1] Reserved position defaults to 000 selection<br>Reset type: CPU1.SYSRSn |
| 8-6 | EPWM10SYNCIN | R/W | 0h | Selects Sync Input Source for EPWM10:<br>000: EPWM1SYNCOUT selected<br>001: EPWM4SYNCOUT selected<br>010: EPWM7SYNCOUT selected<br>011: EPWM10SYNCOUT selected (Reserved)<br>100: ECAP1SYNCOUT selected (Reserved)<br>101: EXTSYNCIN1 selected<br>110: EXTSYNCIN2 selected<br>111: Reserved<br>Notes:<br>[1] Reserved position defaults to 000 selection<br>Reset type: CPU1.SYSRSn |
| 5-3 | EPWM7SYNCIN | R/W | 0h | Selects Sync Input Source for EPWM7:<br>000: EPWM1SYNCOUT selected<br>001: EPWM4SYNCOUT selected<br>010: EPWM7SYNCOUT selected (Reserved)<br>011: EPWM10SYNCOUT selected (Reserved)<br>100: ECAP1SYNCOUT selected (Reserved)<br>101: EXTSYNCIN1 selected<br>110: EXTSYNCIN2 selected<br>111: Reserved<br>Notes:<br>[1] Reserved position defaults to 000 selection<br>Reset type: CPU1.SYSRSn |
| 2-0 | EPWM4SYNCIN | R/W | 0h | Selects Sync Input Source for EPWM4:<br>000: EPWM1SYNCOUT selected<br>001: EPWM4SYNCOUT selected (Reserved)<br>010: EPWM7SYNCOUT selected (Reserved)<br>011: EPWM10SYNCOUT selected (Reserved)<br>100: ECAP1SYNCOUT selected (Reserved)<br>101: EXTSYNCIN1 selected<br>110: EXTSYNCIN2 selected<br>111: Reserved<br>Notes:<br>[1] Reserved position defaults to 000 selection<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.7.2 ADCSOCOUTSELECT Register (Offset = 2h) [Reset = 00000000h]

ADCSOCOUTSELECT is shown in Figure 3-78 and described in Table 3-85.

Return to the Summary Table.

The ADCSOCAO and ADCSOCBO signals will be active low for 32 SYSCLK cycles. They can be used to trigger a conversion on an external ADC.

**Figure 3-78. ADCSOCOUTSELECT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | PWM12SOCBEN | PWM11SOCBEN | PWM10SOCBEN | PWM9SOCBEN |
| R-0-0h | | | | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| PWM8SOCBEN | 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 | | | | PWM12SOCAEN | PWM11SOCAEN | PWM10SOCAEN | PWM9SOCAEN |
| R-0-0h | | | | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PWM8SOCAEN | 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 3-85. ADCSOCOUTSELECT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-28 | RESERVED | R-0 | 0h | Reserved |
| 27 | PWM12SOCBEN | R/W | 0h | ADCSOCBO source select:<br>0: Respective EPWM SOCB output is not selected<br>1: Respective EPWM SOCB output is selected<br>Reset type: CPU1.SYSRSn |
| 26 | PWM11SOCBEN | R/W | 0h | ADCSOCBO source select:<br>0: Respective EPWM SOCB output is not selected<br>1: Respective EPWM SOCB output is selected<br>Reset type: CPU1.SYSRSn |
| 25 | PWM10SOCBEN | R/W | 0h | ADCSOCBO source select:<br>0: Respective EPWM SOCB output is not selected<br>1: Respective EPWM SOCB output is selected<br>Reset type: CPU1.SYSRSn |
| 24 | PWM9SOCBEN | R/W | 0h | ADCSOCBO source select:<br>0: Respective EPWM SOCB output is not selected<br>1: Respective EPWM SOCB output is selected<br>Reset type: CPU1.SYSRSn |
| 23 | PWM8SOCBEN | R/W | 0h | ADCSOCBO source select:<br>0: Respective EPWM SOCB output is not selected<br>1: Respective EPWM SOCB output is selected<br>Reset type: CPU1.SYSRSn |
| 22 | PWM7SOCBEN | R/W | 0h | ADCSOCBO source select:<br>0: Respective EPWM SOCB output is not selected<br>1: Respective EPWM SOCB output is selected<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-85. ADCSOCOUTSELECT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 21 | PWM6SOCBEN | R/W | 0h | ADCSOCBO source select:<br>0: Respective EPWM SOCB output is not selected<br>1: Respective EPWM SOCB output is selected<br>Reset type: CPU1.SYSRSn |
| 20 | PWM5SOCBEN | R/W | 0h | ADCSOCBO source select:<br>0: Respective EPWM SOCB output is not selected<br>1: Respective EPWM SOCB output is selected<br>Reset type: CPU1.SYSRSn |
| 19 | PWM4SOCBEN | R/W | 0h | ADCSOCBO source select:<br>0: Respective EPWM SOCB output is not selected<br>1: Respective EPWM SOCB output is selected<br>Reset type: CPU1.SYSRSn |
| 18 | PWM3SOCBEN | R/W | 0h | ADCSOCBO source select:<br>0: Respective EPWM SOCB output is not selected<br>1: Respective EPWM SOCB output is selected<br>Reset type: CPU1.SYSRSn |
| 17 | PWM2SOCBEN | R/W | 0h | ADCSOCBO source select:<br>0: Respective EPWM SOCB output is not selected<br>1: Respective EPWM SOCB output is selected<br>Reset type: CPU1.SYSRSn |
| 16 | PWM1SOCBEN | R/W | 0h | ADCSOCBO source select:<br>0: Respective EPWM SOCB output is not selected<br>1: Respective EPWM SOCB output is selected<br>Reset type: CPU1.SYSRSn |
| 15-12 | RESERVED | R-0 | 0h | Reserved |
| 11 | PWM12SOCAEN | R/W | 0h | ADCSOCAO source select:<br>0: Respective EPWM SOCA output is not selected<br>1: Respective EPWM SOCA output is selected<br>Reset type: CPU1.SYSRSn |
| 10 | PWM11SOCAEN | R/W | 0h | ADCSOCAO source select:<br>0: Respective EPWM SOCA output is not selected<br>1: Respective EPWM SOCA output is selected<br>Reset type: CPU1.SYSRSn |
| 9 | PWM10SOCAEN | R/W | 0h | ADCSOCAO source select:<br>0: Respective EPWM SOCA output is not selected<br>1: Respective EPWM SOCA output is selected<br>Reset type: CPU1.SYSRSn |
| 8 | PWM9SOCAEN | R/W | 0h | ADCSOCAO source select:<br>0: Respective EPWM SOCA output is not selected<br>1: Respective EPWM SOCA output is selected<br>Reset type: CPU1.SYSRSn |
| 7 | PWM8SOCAEN | R/W | 0h | ADCSOCAO source select:<br>0: Respective EPWM SOCA output is not selected<br>1: Respective EPWM SOCA output is selected<br>Reset type: CPU1.SYSRSn |
| 6 | PWM7SOCAEN | R/W | 0h | ADCSOCAO source select:<br>0: Respective EPWM SOCA output is not selected<br>1: Respective EPWM SOCA output is selected<br>Reset type: CPU1.SYSRSn |
| 5 | PWM6SOCAEN | R/W | 0h | ADCSOCAO source select:<br>0: Respective EPWM SOCA output is not selected<br>1: Respective EPWM SOCA output is selected<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-85. ADCSOCOUTSELECT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 4 | PWM5SOCAEN | R/W | 0h | ADCSOCAO source select:<br>0: Respective EPWM SOCA output is not selected<br>1: Respective EPWM SOCA output is selected<br>Reset type: CPU1.SYSRSn |
| 3 | PWM4SOCAEN | R/W | 0h | ADCSOCAO source select:<br>0: Respective EPWM SOCA output is not selected<br>1: Respective EPWM SOCA output is selected<br>Reset type: CPU1.SYSRSn |
| 2 | PWM3SOCAEN | R/W | 0h | ADCSOCAO source select:<br>0: Respective EPWM SOCA output is not selected<br>1: Respective EPWM SOCA output is selected<br>Reset type: CPU1.SYSRSn |
| 1 | PWM2SOCAEN | R/W | 0h | ADCSOCAO source select:<br>0: Respective EPWM SOCA output is not selected<br>1: Respective EPWM SOCA output is selected<br>Reset type: CPU1.SYSRSn |
| 0 | PWM1SOCAEN | R/W | 0h | ADCSOCAO source select:<br>0: Respective EPWM SOCA output is not selected<br>1: Respective EPWM SOCA output is selected<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.7.3 SYNCSOCLOCK Register (Offset = 4h) [Reset = 00000000h]

SYNCSOCLOCK is shown in Figure 3-79 and described in Table 3-86.

Return to the Summary Table.

SYNCSEL and EXTADCSOC Select Lock register

**Figure 3-79. 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 | | | | | | ADCSOCOUTSELECT | SYNCSELECT |
| R-0-0h | | | | | | R/WSonce-0h | R/WSonce-0h |

**Table 3-86. SYNCSOCLOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-2 | RESERVED | R-0 | 0h | Reserved |
| 1 | ADCSOCOUTSELECT | R/WSonce | 0h | ADCSOCOUTSELECT Register Lock bit:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Notes:<br>[1] Any bit in this register, once set can only be creaed through a CPU1.SYSRSn. Write of 0 to any bit of this regtister has no effect<br>[2] The locking mechanism applies to only writes. Reads to the registers which have LOCK protection are always allowed<br>Reset type: CPU1.SYSRSn |
| 0 | SYNCSELECT | R/WSonce | 0h | SYNCSELECT Register Lock bit:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Notes:<br>[1] Any bit in this register, once set can only be creaed through a CPU1.SYSRSn. Write of 0 to any bit of this regtister has no effect<br>[2] The locking mechanism applies to only writes. Reads to the registers which have LOCK protection are always allowed<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.8 DMA_CLA_SRC_SEL_REGS Registers

Table 3-87 lists the memory-mapped registers for the DMA_CLA_SRC_SEL_REGS registers. All register offset addresses not listed in Table 3-87 should be considered as reserved locations and the register contents should not be modified.

**Table 3-87. DMA_CLA_SRC_SEL_REGS Registers**

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

Complex bit access types are encoded to fit into small table cells. Table 3-88 shows the codes that are used for access types in this section.

**Table 3-88. DMA_CLA_SRC_SEL_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| R-0 | R<br>-0 | Read<br>Returns 0s |
| Write Type | | |
| W | W | Write |
| WSonce | W<br>Sonce | Write<br>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. |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.8.1 CLA1TASKSRCSELLOCK Register (Offset = 0h) [Reset = 00000000h]

CLA1TASKSRCSELLOCK is shown in Figure 3-80 and described in Table 3-89.

Return to the Summary Table.

CLA1 Task Trigger Source Select Lock Register

**Figure 3-80. CLA1TASKSRCSELLOCK Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | CLA1TASKSRCSEL2 | CLA1TASKSRCSEL1 |
| R-0-0h | | | | | | R/WSonce-0h | R/WSonce-0h |

**Table 3-89. CLA1TASKSRCSELLOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-2 | RESERVED | R-0 | 0h | Reserved |
| 1 | CLA1TASKSRCSEL2 | R/WSonce | 0h | CLA1TASKSRCSEL2 Register Lock bit:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Notes:<br>[1] Any SOnce bit in this register, once set can only be cleared through a SYSRSn. Write of 0 to any bit of this register has no effect<br>[2] The locking mechanism applies to only writes. Reads to the registers which have LOCK protection are always allowed<br>Reset type: SYSRSn |
| 0 | CLA1TASKSRCSEL1 | R/WSonce | 0h | CLA1TASKSRCSEL1 Register Lock bit:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Notes:<br>[1] Any SOnce bit in this register, once set can only be cleared through a SYSRSn. Write of 0 to any bit of this register has no effect<br>[2] The locking mechanism applies to only writes. Reads to the registers which have LOCK protection are always allowed<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.8.2 DMACHSRCSELLOCK Register (Offset = 4h) [Reset = 00000000h]

DMACHSRCSELLOCK is shown in Figure 3-81 and described in Table 3-90.

Return to the Summary Table.

DMA Channel Triger Source Select Lock Register

**Figure 3-81. DMACHSRCSELLOCK Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | DMACHSRCSEL2 | DMACHSRCSEL1 |
| R-0-0h | | | | | | R/WSonce-0h | R/WSonce-0h |

**Table 3-90. DMACHSRCSELLOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-2 | RESERVED | R-0 | 0h | Reserved |
| 1 | DMACHSRCSEL2 | R/WSonce | 0h | DMACHSRCSEL2 Register Lock bit:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Notes:<br>[1] Any SOnce bit in this register, once set can only be cleared through a SYSRSn. Write of 0 to any bit of this register has no effect<br>[2] The locking mechanism applies to only writes. Reads to the registers which have LOCK protection are always allowed<br>Reset type: SYSRSn |
| 0 | DMACHSRCSEL1 | R/WSonce | 0h | DMACHSRCSEL1 Register Lock bit:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Notes:<br>[1] Any SOnce bit in this register, once set can only be cleared through a SYSRSn. Write of 0 to any bit of this register has no effect<br>[2] The locking mechanism applies to only writes. Reads to the registers which have LOCK protection are always allowed<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.8.3 CLA1TASKSRCSEL1 Register (Offset = 6h) [Reset = 00000000h]

CLA1TASKSRCSEL1 is shown in Figure 3-82 and described in Table 3-91.

Return to the Summary Table.

CLA1 Task Trigger Source Select Register-1

**Figure 3-82. CLA1TASKSRCSEL1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TASK4 | | | | | | | | TASK3 | | | | | | | | TASK2 | | | | | | | | TASK1 | | | | | | | |
| R/W-0h | | | | | | | | R/W-0h | | | | | | | | R/W-0h | | | | | | | | R/W-0h | | | | | | | |

**Table 3-91. CLA1TASKSRCSEL1 Register Field Descriptions**

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



Copyright © 2024 Texas Instruments Incorporated

### 3.17.8.4 CLA1TASKSRCSEL2 Register (Offset = 8h) [Reset = 00000000h]

CLA1TASKSRCSEL2 is shown in Figure 3-83 and described in Table 3-92.

Return to the Summary Table.

CLA1 Task Trigger Source Select Register-2

**Figure 3-83. CLA1TASKSRCSEL2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TASK8 | | | | | | | | TASK7 | | | | | | | | TASK6 | | | | | | | | TASK5 | | | | | | | |
| R/W-0h | | | | | | | | R/W-0h | | | | | | | | R/W-0h | | | | | | | | R/W-0h | | | | | | | |

**Table 3-92. CLA1TASKSRCSEL2 Register Field Descriptions**

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



Copyright © 2024 Texas Instruments Incorporated

### 3.17.8.5 DMACHSRCSEL1 Register (Offset = 16h) [Reset = 00000000h]

DMACHSRCSEL1 is shown in Figure 3-84 and described in Table 3-93.

Return to the Summary Table.

DMA Channel Trigger Source Select Register-1

**Figure 3-84. DMACHSRCSEL1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH4 | | | | | | | | CH3 | | | | | | | | CH2 | | | | | | | | CH1 | | | | | | | |
| R/W-0h | | | | | | | | R/W-0h | | | | | | | | R/W-0h | | | | | | | | R/W-0h | | | | | | | |

**Table 3-93. DMACHSRCSEL1 Register Field Descriptions**

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



Copyright © 2024 Texas Instruments Incorporated

### 3.17.8.6 DMACHSRCSEL2 Register (Offset = 18h) [Reset = 00000000h]

DMACHSRCSEL2 is shown in Figure 3-85 and described in Table 3-94.

Return to the Summary Table.

DMA Channel Trigger Source Select Register-2

**Figure 3-85. DMACHSRCSEL2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | | CH6 | | | | | | | | CH5 | | | | | | | |
| R-0-0h | | | | | | | | | | | | | | | | R/W-0h | | | | | | | | R/W-0h | | | | | | | |

**Table 3-94. DMACHSRCSEL2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-8 | CH6 | R/W | 0h | Selects the Trigger and Sync Source CH6 of DMA<br>Reset type: SYSRSn |
| 7-0 | CH5 | R/W | 0h | Selects the Trigger and Sync Source CH5 of DMA<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.9 FLASH_PUMP_SEMAPHORE_REGS Registers

Table 3-95 lists the memory-mapped registers for the FLASH_PUMP_SEMAPHORE_REGS registers. All register offset addresses not listed in Table 3-95 should be considered as reserved locations and the register contents should not be modified.

**Table 3-95. FLASH_PUMP_SEMAPHORE_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | PUMPREQUEST | Flash programming semaphore PUMP request register | EALLOW | Go |

Complex bit access types are encoded to fit into small table cells. Table 3-96 shows the codes that are used for access types in this section.

**Table 3-96. FLASH_PUMP_SEMAPHORE_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| R-0 | R<br>-0 | Read<br>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. |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.9.1 PUMPREQUEST Register (Offset = 0h) [Reset = 00000000h]

PUMPREQUEST is shown in Figure 3-86 and described in Table 3-97.

Return to the Summary Table.

Flash programming semaphore PUMP request register

**Figure 3-86. PUMPREQUEST 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 | | | | | | PUMP_OWNERSHIP | |
| R-0-0h | | | | | | R/W-0h | |

**Table 3-97. PUMPREQUEST Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | KEY | R-0/W | 0h | In order to write to the semaphore bits, 0x5a5a must be written to these key bits at the same time. Otherwise, writes are ignored. The key is cleared immediately after writing, so it must be written again for every semaphore change.<br>Reset type: CPU1.SYSRSn |
| 15-2 | RESERVED | R-0 | 0h | Reserved |
| 1-0 | PUMP_OWNERSHIP | R/W | 0h | These bits configure which CPU has control of the flash pump, which allows write access to the flash memory. The possible values are:<br>00 or 11: Read-only state. CPU1 has control of the pump, but CPU2 may seize control at any time.<br>01: CPU2 has exclusive control of the pump and of these semaphore bits. CPU2 can relinquish control by setting the bits back to 00 or 11.<br>10: CPU1 has exclusive control of the pump and of these semaphore bits. CPU1 can relinquish control by setting the bits back to 00 or 11.<br>Going from 01->10 or 10->01 is not allowed. Going from 00->11 or 11->00 is allowed, but has no effect. The semaphore bits [1:0] must be written along with the correct key in bits [31:16].<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10 DEV_CFG_REGS Registers

Table 3-98 lists the memory-mapped registers for the DEV_CFG_REGS registers. All register offset addresses not listed in Table 3-98 should be considered as reserved locations and the register contents should not be modified.

**Table 3-98. DEV_CFG_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | DEVCFGLOCK1 | Lock bit for CPUSELx registers | EALLOW | Go |
| 8h | PARTIDL | Lower 32-bit of Device PART Identification Number | | Go |
| Ah | PARTIDH | Upper 32-bit of Device PART Identification Number | | Go |
| Ch | REVID | Device Revision Number | | Go |
| 10h | DC0 | Device Capability: Device Information | | Go |
| 12h | DC1 | Device Capability: Processing Block Customization | | Go |
| 14h | DC2 | Device Capability: EMIF Customization | | Go |
| 16h | DC3 | Device Capability: Peripheral Customization | | Go |
| 18h | DC4 | Device Capability: Peripheral Customization | | Go |
| 1Ah | DC5 | Device Capability: Peripheral Customization | | Go |
| 1Ch | DC6 | Device Capability: Peripheral Customization | | Go |
| 1Eh | DC7 | Device Capability: Peripheral Customization | | Go |
| 20h | DC8 | Device Capability: Peripheral Customization | | Go |
| 22h | DC9 | Device Capability: Peripheral Customization | | Go |
| 24h | DC10 | Device Capability: Peripheral Customization | | Go |
| 26h | DC11 | Device Capability: Peripheral Customization | | Go |
| 28h | DC12 | Device Capability: Peripheral Customization | | Go |
| 2Ah | DC13 | Device Capability: Peripheral Customization | | Go |
| 2Ch | DC14 | Device Capability: Analog Modules Customization | | Go |
| 2Eh | DC15 | Device Capability: Analog Modules Customization | | Go |
| 32h | DC17 | Device Capability: Analog Modules Customization | | Go |
| 34h | DC18 | Device Capability: CPU1 Lx SRAM Customization | | Go |
| 36h | DC19 | Device Capability: CPU2 Lx SRAM Customization | | Go |
| 38h | DC20 | Device Capability: GSx SRAM Customization | | Go |
| 60h | PERCNF1 | Peripheral Configuration register | | Go |
| 74h | FUSEERR | e-Fuse error Status register | | Go |
| 82h | SOFTPRES0 | Processing Block Software Reset register | EALLOW | Go |
| 84h | SOFTPRES1 | EMIF Software Reset register | EALLOW | Go |
| 86h | SOFTPRES2 | Peripheral Software Reset register | EALLOW | Go |
| 88h | SOFTPRES3 | Peripheral Software Reset register | EALLOW | Go |
| 8Ah | SOFTPRES4 | Peripheral Software Reset register | EALLOW | Go |
| 8Eh | SOFTPRES6 | Peripheral Software Reset register | EALLOW | Go |
| 90h | SOFTPRES7 | Peripheral Software Reset register | EALLOW | Go |
| 92h | SOFTPRES8 | Peripheral Software Reset register | EALLOW | Go |
| 94h | SOFTPRES9 | Peripheral Software Reset register | EALLOW | Go |
| 98h | SOFTPRES11 | Peripheral Software Reset register | EALLOW | Go |
| 9Ch | SOFTPRES13 | Peripheral Software Reset register | EALLOW | Go |
| 9Eh | SOFTPRES14 | Peripheral Software Reset register | EALLOW | Go |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-98. DEV_CFG_REGS Registers (continued)**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| A2h | SOFTPRES16 | Peripheral Software Reset register | EALLOW | Go |
| D6h | CPUSEL0 | CPU Select register for common peripherals | EALLOW | Go |
| D8h | CPUSEL1 | CPU Select register for common peripherals | EALLOW | Go |
| DAh | CPUSEL2 | CPU Select register for common peripherals | EALLOW | Go |
| DCh | CPUSEL3 | CPU Select register for common peripherals | EALLOW | Go |
| DEh | CPUSEL4 | CPU Select register for common peripherals | EALLOW | Go |
| E0h | CPUSEL5 | CPU Select register for common peripherals | EALLOW | Go |
| E2h | CPUSEL6 | CPU Select register for common peripherals | EALLOW | Go |
| E4h | CPUSEL7 | CPU Select register for common peripherals | EALLOW | Go |
| E6h | CPUSEL8 | CPU Select register for common peripherals | EALLOW | Go |
| E8h | CPUSEL9 | CPU Select register for common peripherals | EALLOW | Go |
| ECh | CPUSEL11 | CPU Select register for common peripherals | EALLOW | Go |
| EEh | CPUSEL12 | CPU Select register for common peripherals | EALLOW | Go |
| F2h | CPUSEL14 | CPU Select register for common peripherals | EALLOW | Go |
| 122h | CPU2RESCTL | CPU2 Reset Control Register | EALLOW | Go |
| 124h | RSTSTAT | Reset Status register for secondary C28x CPUs | | Go |
| 125h | LPMSTAT | LPM Status Register for secondary C28x CPUs | | Go |
| 12Ch | SYSDBGCTL | System Debug Control register | EALLOW | Go |

Complex bit access types are encoded to fit into small table cells. Table 3-99 shows the codes that are used for access types in this section.

**Table 3-99. DEV_CFG_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| R-0 | R<br>-0 | Read<br>Returns 0s |
| Write Type | | |
| W | W | Write |
| W1S | W<br>1S | Write<br>1 to set |
| WOnce | W<br>Once | Write<br>Write once |
| WSonce | W<br>Sonce | Write<br>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. |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.1 DEVCFGLOCK1 Register (Offset = 0h) [Reset = 00000000h]

DEVCFGLOCK1 is shown in Figure 3-87 and described in Table 3-100.

Return to the Summary Table.

Lock bit for CPUSELx registers
The locking mechanism applies to only writes. Reads to the registers which have LOCK protection are always allowed
Note:
Any SOnce bit in this register, once set can only be cleared through a CPU1.SYSRSn. Write of 0 to any bit of this register has no effect

**Figure 3-87. DEVCFGLOCK1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | CPUSEL14 | CPUSEL13 | CPUSEL12 | CPUSEL11 | CPUSEL10 | CPUSEL9 | CPUSEL8 |
| R-0-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CPUSEL7 | CPUSEL6 | CPUSEL5 | CPUSEL4 | CPUSEL3 | CPUSEL2 | CPUSEL1 | CPUSEL0 |
| R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h |

**Table 3-100. DEVCFGLOCK1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15 | RESERVED | R-0 | 0h | Reserved |
| 14 | CPUSEL14 | R/WSonce | 0h | Lock bit for CPUSEL14 register:<br>0: Register is not locked<br>1: Register is locked<br>Reset type: CPU1.SYSRSn |
| 13 | CPUSEL13 | R/WSonce | 0h | Lock bit for CPUSEL13 register:<br>0: Register is not locked<br>1: Register is locked<br>Reset type: CPU1.SYSRSn |
| 12 | CPUSEL12 | R/WSonce | 0h | Lock bit for CPUSEL12 register:<br>0: Register is not locked<br>1: Register is locked<br>Reset type: CPU1.SYSRSn |
| 11 | CPUSEL11 | R/WSonce | 0h | Lock bit for CPUSEL11 register:<br>0: Register is not locked<br>1: Register is locked<br>Reset type: CPU1.SYSRSn |
| 10 | CPUSEL10 | R/WSonce | 0h | Lock bit for CPUSEL10 register:<br>0: Register is not locked<br>1: Register is locked<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-100. DEVCFGLOCK1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 9 | CPUSEL9 | R/WSonce | 0h | Lock bit for CPUSEL9 register:<br>0: Register is not locked<br>1: Register is locked<br>Reset type: CPU1.SYSRSn |
| 8 | CPUSEL8 | R/WSonce | 0h | Lock bit for CPUSEL8 register:<br>0: Register is not locked<br>1: Register is locked<br>Reset type: CPU1.SYSRSn |
| 7 | CPUSEL7 | R/WSonce | 0h | Lock bit for CPUSEL7 register:<br>0: Register is not locked<br>1: Register is locked<br>Reset type: CPU1.SYSRSn |
| 6 | CPUSEL6 | R/WSonce | 0h | Lock bit for CPUSEL6 register:<br>0: Register is not locked<br>1: Register is locked<br>Reset type: CPU1.SYSRSn |
| 5 | CPUSEL5 | R/WSonce | 0h | Lock bit for CPUSEL5 register:<br>0: Register is not locked<br>1: Register is locked<br>Reset type: CPU1.SYSRSn |
| 4 | CPUSEL4 | R/WSonce | 0h | Lock bit for CPUSEL4 register:<br>0: Register is not locked<br>1: Register is locked<br>Reset type: CPU1.SYSRSn |
| 3 | CPUSEL3 | R/WSonce | 0h | Lock bit for CPUSEL3 register:<br>0: Register is not locked<br>1: Register is locked<br>Reset type: CPU1.SYSRSn |
| 2 | CPUSEL2 | R/WSonce | 0h | Lock bit for CPUSEL2 register:<br>0: Register is not locked<br>1: Register is locked<br>Reset type: CPU1.SYSRSn |
| 1 | CPUSEL1 | R/WSonce | 0h | Lock bit for CPUSEL1 register:<br>0: Register is not locked<br>1: Register is locked<br>Reset type: CPU1.SYSRSn |
| 0 | CPUSEL0 | R/WSonce | 0h | Lock bit for CPUSEL0 register:<br>0: Register is not locked<br>1: Register is locked<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.2 PARTIDL Register (Offset = 8h) [Reset = 00000000h]

PARTIDL is shown in Figure 3-88 and described in Table 3-101.

Return to the Summary Table.

Lower 32-bit of Device PART Identification Number

**Figure 3-88. PARTIDL Register**

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

**Table 3-101. PARTIDL Register Field Descriptions**

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



Copyright © 2024 Texas Instruments Incorporated

**Table 3-101. PARTIDL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 2-0 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.3 PARTIDH Register (Offset = Ah) [Reset = 00000000h]

PARTIDH is shown in Figure 3-89 and described in Table 3-102.

Return to the Summary Table.

Upper 32-bit of Device PART Identification Number

**Figure 3-89. PARTIDH Register**

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

**Table 3-102. PARTIDH Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-24 | DEVICE_CLASS_ID | R | 0h | Reserved<br>Reset type: XRSn |
| 23-16 | PARTNO | R | 0h | Refer to Datasheet for Device Part Number<br>Reset type: XRSn |
| 15-8 | FAMILY | R | 0h | Device Family<br>0x3 - DUAL CORE<br>0x4 - SINGLE CORE<br>0x5 - PICCOLO SINGLE CORE<br>Other values Reserved<br>Reset type: XRSn |
| 7-0 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.4 REVID Register (Offset = Ch) [Reset = 00000000h]

REVID is shown in Figure 3-90 and described in Table 3-103.

Return to the Summary Table.

Device Revision Number

**Figure 3-90. REVID Register**

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

**Table 3-103. REVID Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | REVID | R | 0h | These 32-bits specify the silicon revision. See your device specific datasheet for details.<br>Reset type: N/A |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.5 DC0 Register (Offset = 10h) [Reset = 0000000Xh]

DC0 is shown in Figure 3-91 and described in Table 3-104.

Return to the Summary Table.

Device Capability: Device Information

**Figure 3-91. DC0 Register**

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

**Table 3-104. DC0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-1 | RESERVED | R-0 | 0h | Reserved |
| 0 | SINGLE_CORE | R | X | Single Core vs Dual Core<br>0: Single Core<br>1: Dual Core<br>Reset type: XRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.6 DC1 Register (Offset = 12h) [Reset = 00000XXXh]

DC1 is shown in Figure 3-92 and described in Table 3-105.

Return to the Summary Table.

Device Capability: Processing Block Customization

**Figure 3-92. DC1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | RESERVED | CPU2_CLA1 |
| R-0-0h | | | | | | R-X | R-X |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | CPU1_CLA1 | RESERVED | | CPU2_VCU | CPU1_VCU | CPU2_FPU_TMU | CPU1_FPU_TMU |
| R-X | R-X | R-0-0h | | R-X | R-X | R-X | R-X |

**Table 3-105. DC1 Register Field Descriptions**

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



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.7 DC2 Register (Offset = 14h) [Reset = 0000000Xh]

DC2 is shown in Figure 3-93 and described in Table 3-106.

Return to the Summary Table.

Device Capability: EMIF Customization

**Figure 3-93. DC2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | EMIF2 | EMIF1 |
| R-0-0h | | | | | | R-X | R-X |

**Table 3-106. DC2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-2 | RESERVED | R-0 | 0h | Reserved |
| 1 | EMIF2 | R | X | EMIF2 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 0 | EMIF1 | R | X | EMIF1 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.8 DC3 Register (Offset = 16h) [Reset = 0000XXXXh]

DC3 is shown in Figure 3-94 and described in Table 3-107.

Return to the Summary Table.

Device Capability: Peripheral Customization

**Figure 3-94. DC3 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | RESERVED | RESERVED | RESERVED | EPWM12 | EPWM11 | EPWM10 | EPWM9 |
| R-X | R-X | R-X | R-X | R-X | R-X | R-X | R-X |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EPWM8 | EPWM7 | EPWM6 | EPWM5 | EPWM4 | EPWM3 | EPWM2 | EPWM1 |
| R-X | R-X | R-X | R-X | R-X | R-X | R-X | R-X |

**Table 3-107. DC3 Register Field Descriptions**

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



Copyright © 2024 Texas Instruments Incorporated

**Table 3-107. DC3 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 5 | EPWM6 | R | X | EPWM6 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 4 | EPWM5 | R | X | EPWM5 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 3 | EPWM4 | R | X | EPWM4 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 2 | EPWM3 | R | X | EPWM3 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 1 | EPWM2 | R | X | EPWM2 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 0 | EPWM1 | R | X | EPWM1 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.9 DC4 Register (Offset = 18h) [Reset = 000000XXh]

DC4 is shown in Figure 3-95 and described in Table 3-108.

Return to the Summary Table.

Device Capability: Peripheral Customization

**Figure 3-95. DC4 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | RESERVED | ECAP6 | ECAP5 | ECAP4 | ECAP3 | ECAP2 | ECAP1 |
| R-X | R-X | R-X | R-X | R-X | R-X | R-X | R-X |

**Table 3-108. DC4 Register Field Descriptions**

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



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.10 DC5 Register (Offset = 1Ah) [Reset = 0000000Xh]

DC5 is shown in Figure 3-96 and described in Table 3-109.

Return to the Summary Table.

Device Capability: Peripheral Customization

**Figure 3-96. DC5 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | RESERVED | EQEP3 | EQEP2 | EQEP1 |
| R-0-0h | | | | R-X | R-X | R-X | R-X |

**Table 3-109. DC5 Register Field Descriptions**

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



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.11 DC6 Register (Offset = 1Ch) [Reset = 000000XXh]

DC6 is shown in Figure 3-97 and described in Table 3-110.

Return to the Summary Table.

Device Capability: Peripheral Customization

**Figure 3-97. DC6 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | RESERVED | RESERVED | RESERVED | CLB4 | CLB3 | CLB2 | CLB1 |
| R-X | R-X | R-X | R-X | R-X | R-X | R-X | R-X |

**Table 3-110. DC6 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-8 | RESERVED | R-0 | 0h | Reserved |
| 7 | RESERVED | R | X | Reserved |
| 6 | RESERVED | R | X | Reserved |
| 5 | RESERVED | R | X | Reserved |
| 4 | RESERVED | R | X | Reserved |
| 3 | CLB4 | R | X | CLB4 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 2 | CLB3 | R | X | CLB3 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 1 | CLB2 | R | X | CLB2 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 0 | CLB1 | R | X | CLB1 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.12 DC7 Register (Offset = 1Eh) [Reset = 000000XXh]

DC7 is shown in Figure 3-98 and described in Table 3-111.

Return to the Summary Table.

Device Capability: Peripheral Customization

**Figure 3-98. DC7 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | |
| R-0-0h | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | | | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | SD2 | SD1 |
| R-0-0h | | | | | | | | R-X | R-X | R-X | R-X | R-X | R-X | R-X | R-X |

**Table 3-111. DC7 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-8 | RESERVED | R-0 | 0h | Reserved |
| 7 | RESERVED | R | X | Reserved |
| 6 | RESERVED | R | X | Reserved |
| 5 | RESERVED | R | X | Reserved |
| 4 | RESERVED | R | X | Reserved |
| 3 | RESERVED | R | X | Reserved |
| 2 | RESERVED | R | X | Reserved |
| 1 | SD2 | R | X | SD2 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 0 | SD1 | R | X | SD1 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.13 DC8 Register (Offset = 20h) [Reset = 0000000Xh]

DC8 is shown in Figure 3-99 and described in Table 3-112.

Return to the Summary Table.

Device Capability: Peripheral Customization

**Figure 3-99. DC8 Register**

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

**Table 3-112. DC8 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-4 | RESERVED | R-0 | 0h | Reserved |
| 3 | SCI_D | R | X | SCI_D :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 2 | SCI_C | R | X | SCI_C :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 1 | SCI_B | R | X | SCI_B :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 0 | SCI_A | R | X | SCI_A :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.14 DC9 Register (Offset = 22h) [Reset = 000X000Xh]

DC9 is shown in Figure 3-100 and described in Table 3-113.

Return to the Summary Table.

Device Capability: Peripheral Customization

**Figure 3-100. DC9 Register**

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

**Table 3-113. DC9 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-18 | RESERVED | R-0 | 0h | Reserved |
| 17 | RESERVED | R | X | Reserved |
| 16 | RESERVED | R | X | Reserved |
| 15-4 | RESERVED | R-0 | 0h | Reserved |
| 3 | RESERVED | R | X | Reserved |
| 2 | SPI_C | R | X | SPI_C :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 1 | SPI_B | R | X | SPI_B :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 0 | SPI_A | R | X | SPI_A :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.15 DC10 Register (Offset = 24h) [Reset = 000X000Xh]

DC10 is shown in Figure 3-101 and described in Table 3-114.

Return to the Summary Table.

Device Capability: Peripheral Customization

**Figure 3-101. DC10 Register**

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

**Table 3-114. DC10 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-18 | RESERVED | R-0 | 0h | Reserved |
| 17 | RESERVED | R | X | Reserved |
| 16 | RESERVED | R | X | Reserved |
| 15-2 | RESERVED | R-0 | 0h | Reserved |
| 1 | I2C_B | R | X | I2C_B :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 0 | I2C_A | R | X | I2C_A :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.16 DC11 Register (Offset = 26h) [Reset = 0000000Xh]

DC11 is shown in Figure 3-102 and described in Table 3-115.

Return to the Summary Table.

Device Capability: Peripheral Customization

**Figure 3-102. DC11 Register**

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

**Table 3-115. DC11 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-4 | RESERVED | R-0 | 0h | Reserved |
| 3 | RESERVED | R | X | Reserved |
| 2 | RESERVED | R | X | Reserved |
| 1 | CAN_B | R | X | CAN_B :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 0 | CAN_A | R | X | CAN_A :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.17 DC12 Register (Offset = 28h) [Reset = 000X000Xh]

DC12 is shown in Figure 3-103 and described in Table 3-116.

Return to the Summary Table.

Device Capability: Peripheral Customization

**Figure 3-103. DC12 Register**

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

**Table 3-116. DC12 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-20 | RESERVED | R-0 | 0h | Reserved |
| 19-18 | RESERVED | R | X | Reserved |
| 17-16 | USB_A | R | X | Capability of the USB_A Module:<br>2'b00: No USB function<br>2'b01: Device Only<br>2'b10: Device or Host<br>2'b11: Device or Host<br>Reset type: XRSn |
| 15-2 | RESERVED | R-0 | 0h | Reserved |
| 1 | McBSP_B | R | X | McBSP_B :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 0 | McBSP_A | R | X | McBSP_A :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.18 DC13 Register (Offset = 2Ah) [Reset = 0000000Xh]

DC13 is shown in Figure 3-104 and described in Table 3-117.

Return to the Summary Table.

Device Capability: Peripheral Customization

**Figure 3-104. DC13 Register**

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

**Table 3-117. DC13 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-2 | RESERVED | R-0 | 0h | Reserved |
| 1 | RESERVED | R | X | Reserved |
| 0 | uPP_A | R | X | uPP_A :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.19 DC14 Register (Offset = 2Ch) [Reset = 0000000Xh]

DC14 is shown in Figure 3-105 and described in Table 3-118.

Return to the Summary Table.

Device Capability: Analog Modules Customization

**Figure 3-105. DC14 Register**

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

**Table 3-118. DC14 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-4 | RESERVED | R-0 | 0h | Reserved |
| 3 | ADC_D | R | X | ADC_D :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 2 | ADC_C | R | X | ADC_C :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 1 | ADC_B | R | X | ADC_B :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 0 | ADC_A | R | X | ADC_A :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.20 DC15 Register (Offset = 2Eh) [Reset = 000000XXh]

DC15 is shown in Figure 3-106 and described in Table 3-119.

Return to the Summary Table.

Device Capability: Analog Modules Customization

**Figure 3-106. DC15 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CMPSS8 | CMPSS7 | CMPSS6 | CMPSS5 | CMPSS4 | CMPSS3 | CMPSS2 | CMPSS1 |
| R-X | R-X | R-X | R-X | R-X | R-X | R-X | R-X |

**Table 3-119. DC15 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-8 | RESERVED | R-0 | 0h | Reserved |
| 7 | CMPSS8 | R | X | CMPSS8 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 6 | CMPSS7 | R | X | CMPSS7 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 5 | CMPSS6 | R | X | CMPSS6 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 4 | CMPSS5 | R | X | CMPSS5 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 3 | CMPSS4 | R | X | CMPSS4 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 2 | CMPSS3 | R | X | CMPSS3 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 1 | CMPSS2 | R | X | CMPSS2 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-119. DC15 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | CMPSS1 | R | X | CMPSS1 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.21 DC17 Register (Offset = 32h) [Reset = 000X000Xh]

DC17 is shown in Figure 3-107 and described in Table 3-120.

Return to the Summary Table.

Device Capability: Analog Modules Customization

**Figure 3-107. DC17 Register**

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

**Table 3-120. DC17 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-20 | RESERVED | R-0 | 0h | Reserved |
| 19 | RESERVED | R | X | Reserved |
| 18 | DAC_C | R | X | Buffered-DAC_C :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 17 | DAC_B | R | X | Buffered-DAC_B :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 16 | DAC_A | R | X | Buffered-DAC_A :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 15-4 | RESERVED | R-0 | 0h | Reserved |
| 3 | RESERVED | R | X | Reserved |
| 2 | RESERVED | R | X | Reserved |
| 1 | RESERVED | R | X | Reserved |
| 0 | RESERVED | R | X | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.22 DC18 Register (Offset = 34h) [Reset = 000000XXh]

DC18 is shown in Figure 3-108 and described in Table 3-121.

Return to the Summary Table.

Device Capability: CPU1 Lx SRAM Customization

**Figure 3-108. DC18 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | LS5_1 | LS4_1 | LS3_1 | LS2_1 | LS1_1 | LS0_1 |
| R-0-0h | | R-X | R-X | R-X | R-X | R-X | R-X |

**Table 3-121. DC18 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-6 | RESERVED | R-0 | 0h | Reserved |
| 5 | LS5_1 | R | X | LS5_1 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 4 | LS4_1 | R | X | LS4_1 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 3 | LS3_1 | R | X | LS3_1 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 2 | LS2_1 | R | X | LS2_1 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 1 | LS1_1 | R | X | LS1_1 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 0 | LS0_1 | R | X | LS0_1 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.23 DC19 Register (Offset = 36h) [Reset = 000000XXh]

DC19 is shown in Figure 3-109 and described in Table 3-122.

Return to the Summary Table.

Device Capability: CPU2 Lx SRAM Customization

**Figure 3-109. DC19 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | LS5_2 | LS4_2 | LS3_2 | LS2_2 | LS1_2 | LS0_2 |
| R-0-0h | | R-X | R-X | R-X | R-X | R-X | R-X |

**Table 3-122. DC19 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-6 | RESERVED | R-0 | 0h | Reserved |
| 5 | LS5_2 | R | X | LS5_2 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 4 | LS4_2 | R | X | LS4_2 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 3 | LS3_2 | R | X | LS3_2 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 2 | LS2_2 | R | X | LS2_2 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 1 | LS1_2 | R | X | LS1_2 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 0 | LS0_2 | R | X | LS0_2 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.24 DC20 Register (Offset = 38h) [Reset = 0000XXXXh]

DC20 is shown in Figure 3-110 and described in Table 3-123.

Return to the Summary Table.

Device Capability: GSx SRAM Customization

**Figure 3-110. DC20 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GS15 | GS14 | GS13 | GS12 | GS11 | GS10 | GS9 | GS8 |
| R-X | R-X | R-X | R-X | R-X | R-X | R-X | R-X |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GS7 | GS6 | GS5 | GS4 | GS3 | GS2 | GS1 | GS0 |
| R-X | R-X | R-X | R-X | R-X | R-X | R-X | R-X |

**Table 3-123. DC20 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15 | GS15 | R | X | GS15 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 14 | GS14 | R | X | GS14 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 13 | GS13 | R | X | GS13 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 12 | GS12 | R | X | GS12 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 11 | GS11 | R | X | GS11 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 10 | GS10 | R | X | GS10 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 9 | GS9 | R | X | GS9 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-123. DC20 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 8 | GS8 | R | X | GS8 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 7 | GS7 | R | X | GS7 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 6 | GS6 | R | X | GS6 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 5 | GS5 | R | X | GS5 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 4 | GS4 | R | X | GS4 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 3 | GS3 | R | X | GS3 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 2 | GS2 | R | X | GS2 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 1 | GS1 | R | X | GS1 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |
| 0 | GS0 | R | X | GS0 :<br>0: Feature not present on the device<br>1: Feature present on the device<br>Reset type: XRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.25 PERCNF1 Register (Offset = 60h) [Reset = 000X000Xh]

PERCNF1 is shown in Figure 3-111 and described in Table 3-124.

Return to the Summary Table.

Peripheral Configuration register

**Figure 3-111. PERCNF1 Register**

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

**Table 3-124. PERCNF1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-18 | RESERVED | R-0 | 0h | Reserved |
| 17 | RESERVED | R | X | Reserved |
| 16 | USB_A_PHY | R | X | Internal PHY is present present or not for the USB_A module:<br>0: Internal USB PHY Module is not present<br>1: Internal USB PHY Module is present.<br>Reset type: XRSn |
| 15-4 | RESERVED | R-0 | 0h | Reserved |
| 3 | ADC_D_MODE | R | X | 0: 16-bit or 12-bit configurable in software<br>1: Only 12-bit operation available<br>Reset type: XRSn |
| 2 | ADC_C_MODE | R | X | 0: 16-bit or 12-bit configurable in software<br>1: Only 12-bit operation available<br>Reset type: XRSn |
| 1 | ADC_B_MODE | R | X | 0: 16-bit or 12-bit configurable in software<br>1: Only 12-bit operation available<br>Reset type: XRSn |
| 0 | ADC_A_MODE | R | X | 0: 16-bit or 12-bit configurable in software<br>1: Only 12-bit operation available<br>Reset type: XRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.26 FUSEERR Register (Offset = 74h) [Reset = 00000000h]

FUSEERR is shown in Figure 3-112 and described in Table 3-125.

Return to the Summary Table.

e-Fuse error Status register

**Figure 3-112. FUSEERR Register**

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

**Table 3-125. FUSEERR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-6 | RESERVED | R-0 | 0h | Reserved |
| 5 | ERR | R | 0h | Efuse Self Test Error Status set by hardware after fuse self test completes, in case of self test error<br>0: No error during fuse self test<br>1: Fuse self test error<br>Reset type: XRSn |
| 4-0 | ALERR | R | 0h | Efuse Autoload Error Status set by hardware after fuse auto load completes<br>00000: No error in auto load<br>Other: Non zero value indicates error in autoload<br>Reset type: XRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.27 SOFTPRES0 Register (Offset = 82h) [Reset = 00000000h]

SOFTPRES0 is shown in Figure 3-113 and described in Table 3-126.

Return to the Summary Table.

Processing Block Software Reset register
When bits in this register are set, the respective module is in reset. All design data is lost and the module registers are returned to their reset states. Bits must be manually cleared after being set.

**Figure 3-113. SOFTPRES0 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | RESERVED | CPU2_CLA1 | RESERVED | CPU1_CLA1 |
| R-0-0h | | | | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-126. SOFTPRES0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-4 | RESERVED | R-0 | 0h | Reserved |
| 3 | RESERVED | R/W | 0h | Reserved |
| 2 | CPU2_CLA1 | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |
| 1 | RESERVED | R/W | 0h | Reserved |
| 0 | CPU1_CLA1 | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.28 SOFTPRES1 Register (Offset = 84h) [Reset = 00000000h]

SOFTPRES1 is shown in Figure 3-114 and described in Table 3-127.

Return to the Summary Table.

EMIF Software Reset register

**Figure 3-114. SOFTPRES1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | EMIF2 | EMIF1 |
| R-0-0h | | | | | | R/W-0h | R/W-0h |

**Table 3-127. SOFTPRES1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-2 | RESERVED | R-0 | 0h | Reserved |
| 1 | EMIF2 | R/W | 0h | When this bit is set, only the control logic of the respective EMIF2 is reset. It does not reset the internal registers except the Total Access register and the Total Activate register.<br>This bit must be manually cleared after being set.<br>1: EMIF2 is under SOFTRESET<br>0: Module reset is determined by the device Reset Network<br>Reset type: CPU1.SYSRSn |
| 0 | EMIF1 | R/W | 0h | When this bit is set, only the control logic of the respective EMIF1 is reset. It does not reset the internal registers except the Total Access register and the Total Activate register.<br>This bit must be manually cleared after being set.<br>1: EMIF1 is under SOFTRESET<br>0: Module reset is determined by the device Reset Network<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.29 SOFTPRES2 Register (Offset = 86h) [Reset = 00000000h]

SOFTPRES2 is shown in Figure 3-115 and described in Table 3-128.

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 3-115. SOFTPRES2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | RESERVED | RESERVED | RESERVED | EPWM12 | EPWM11 | EPWM10 | EPWM9 |
| R/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 |
| EPWM8 | EPWM7 | EPWM6 | EPWM5 | EPWM4 | EPWM3 | EPWM2 | EPWM1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-128. 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 | EPWM12 | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |
| 10 | EPWM11 | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |
| 9 | EPWM10 | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |
| 8 | EPWM9 | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |
| 7 | EPWM8 | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |
| 6 | EPWM7 | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |
| 5 | EPWM6 | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-128. SOFTPRES2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 4 | EPWM5 | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |
| 3 | EPWM4 | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |
| 2 | EPWM3 | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |
| 1 | EPWM2 | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |
| 0 | EPWM1 | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.30 SOFTPRES3 Register (Offset = 88h) [Reset = 00000000h]

SOFTPRES3 is shown in Figure 3-116 and described in Table 3-129.

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 3-116. SOFTPRES3 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | RESERVED | ECAP6 | ECAP5 | ECAP4 | ECAP3 | ECAP2 | ECAP1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-129. SOFTPRES3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-8 | RESERVED | R-0 | 0h | Reserved |
| 7 | RESERVED | R/W | 0h | Reserved |
| 6 | RESERVED | R/W | 0h | Reserved |
| 5 | ECAP6 | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |
| 4 | ECAP5 | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |
| 3 | ECAP4 | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |
| 2 | ECAP3 | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |
| 1 | ECAP2 | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |
| 0 | ECAP1 | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.31 SOFTPRES4 Register (Offset = 8Ah) [Reset = 00000000h]

SOFTPRES4 is shown in Figure 3-117 and described in Table 3-130.

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 3-117. SOFTPRES4 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | RESERVED | EQEP3 | EQEP2 | EQEP1 |
| R-0-0h | | | | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-130. SOFTPRES4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-4 | RESERVED | R-0 | 0h | Reserved |
| 3 | RESERVED | R/W | 0h | Reserved |
| 2 | EQEP3 | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |
| 1 | EQEP2 | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |
| 0 | EQEP1 | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.32 SOFTPRES6 Register (Offset = 8Eh) [Reset = 00000000h]

SOFTPRES6 is shown in Figure 3-118 and described in Table 3-131.

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 3-118. SOFTPRES6 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | |
| R-0-0h | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | | | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | SD2 | SD1 |
| R-0-0h | | | | | | | | 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 3-131. SOFTPRES6 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-8 | RESERVED | R-0 | 0h | Reserved |
| 7 | RESERVED | R/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 | SD2 | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |
| 0 | SD1 | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.33 SOFTPRES7 Register (Offset = 90h) [Reset = 00000000h]

SOFTPRES7 is shown in Figure 3-119 and described in Table 3-132.

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 3-119. SOFTPRES7 Register**

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

**Table 3-132. SOFTPRES7 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-4 | RESERVED | R-0 | 0h | Reserved |
| 3 | SCI_D | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |
| 2 | SCI_C | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |
| 1 | SCI_B | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |
| 0 | SCI_A | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.34 SOFTPRES8 Register (Offset = 92h) [Reset = 00000000h]

SOFTPRES8 is shown in Figure 3-120 and described in Table 3-133.

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 3-120. 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 | SPI_C | SPI_B | SPI_A |
| R-0-0h | | | | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-133. 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 | SPI_C | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |
| 1 | SPI_B | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |
| 0 | SPI_A | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.35 SOFTPRES9 Register (Offset = 94h) [Reset = 00000000h]

SOFTPRES9 is shown in Figure 3-121 and described in Table 3-134.

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 3-121. SOFTPRES9 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | RESERVED | RESERVED |
| R-0-0h | | | | | | R/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 | | | | | | I2C_B | I2C_A |
| R-0-0h | | | | | | R/W-0h | R/W-0h |

**Table 3-134. SOFTPRES9 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 |
| 15-2 | RESERVED | R-0 | 0h | Reserved |
| 1 | I2C_B | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |
| 0 | I2C_A | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.36 SOFTPRES11 Register (Offset = 98h) [Reset = 00000000h]

SOFTPRES11 is shown in Figure 3-122 and described in Table 3-135.

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 3-122. SOFTPRES11 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | RESERVED | USB_A |
| R-0-0h | | | | | | R/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 | | | | | | McBSP_B | McBSP_A |
| R-0-0h | | | | | | R/W-0h | R/W-0h |

**Table 3-135. SOFTPRES11 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-18 | RESERVED | R-0 | 0h | Reserved |
| 17 | RESERVED | R/W | 0h | Reserved |
| 16 | USB_A | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |
| 15-2 | RESERVED | R-0 | 0h | Reserved |
| 1 | McBSP_B | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |
| 0 | McBSP_A | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.37 SOFTPRES13 Register (Offset = 9Ch) [Reset = 00000000h]

SOFTPRES13 is shown in Figure 3-123 and described in Table 3-136.

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 3-123. SOFTPRES13 Register**

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

**Table 3-136. SOFTPRES13 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-4 | RESERVED | R-0 | 0h | Reserved |
| 3 | ADC_D | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |
| 2 | ADC_C | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |
| 1 | ADC_B | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |
| 0 | ADC_A | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.38 SOFTPRES14 Register (Offset = 9Eh) [Reset = 00000000h]

SOFTPRES14 is shown in Figure 3-124 and described in Table 3-137.

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 3-124. SOFTPRES14 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CMPSS8 | CMPSS7 | CMPSS6 | CMPSS5 | CMPSS4 | CMPSS3 | CMPSS2 | CMPSS1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-137. SOFTPRES14 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-8 | RESERVED | R-0 | 0h | Reserved |
| 7 | CMPSS8 | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |
| 6 | CMPSS7 | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |
| 5 | CMPSS6 | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |
| 4 | CMPSS5 | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |
| 3 | CMPSS4 | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |
| 2 | CMPSS3 | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |
| 1 | CMPSS2 | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |
| 0 | CMPSS1 | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.39 SOFTPRES16 Register (Offset = A2h) [Reset = 00000000h]

SOFTPRES16 is shown in Figure 3-125 and described in Table 3-138.

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 3-125. SOFTPRES16 Register**

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

**Table 3-138. SOFTPRES16 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-20 | RESERVED | R-0 | 0h | Reserved |
| 19 | RESERVED | R/W | 0h | Reserved |
| 18 | DAC_C | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |
| 17 | DAC_B | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |
| 16 | DAC_A | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: CPU1.SYSRSn |
| 15-4 | RESERVED | R-0 | 0h | Reserved |
| 3 | RESERVED | R/W | 0h | Reserved |
| 2 | RESERVED | R/W | 0h | Reserved |
| 1 | RESERVED | R/W | 0h | Reserved |
| 0 | RESERVED | R/W | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.40 CPUSEL0 Register (Offset = D6h) [Reset = 00000000h]

CPUSEL0 is shown in Figure 3-126 and described in Table 3-139.

Return to the Summary Table.

CPU Select register for common peripherals
This register must be configured prior to enabling the peripheral clocks.
The clock for each peripheral is derived from the selected CPU subsystem. The clock mux controlled by this register is not glitch-free, therefore the CPUSELx register must be configured before the PCLKCRx register.
The reset for each peripheral is also driven from the selected CPU.

**Figure 3-126. CPUSEL0 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | RESERVED | RESERVED | RESERVED | EPWM12 | EPWM11 | EPWM10 | EPWM9 |
| R/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 |
| EPWM8 | EPWM7 | EPWM6 | EPWM5 | EPWM4 | EPWM3 | EPWM2 | EPWM1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-139. CPUSEL0 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 | EPWM12 | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |
| 10 | EPWM11 | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |
| 9 | EPWM10 | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |
| 8 | EPWM9 | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |
| 7 | EPWM8 | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |
| 6 | EPWM7 | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-139. CPUSEL0 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 5 | EPWM6 | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |
| 4 | EPWM5 | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |
| 3 | EPWM4 | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |
| 2 | EPWM3 | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |
| 1 | EPWM2 | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |
| 0 | EPWM1 | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.41 CPUSEL1 Register (Offset = D8h) [Reset = 00000000h]

CPUSEL1 is shown in Figure 3-127 and described in Table 3-140.

Return to the Summary Table.

CPU Select register for common peripherals
This register must be configured prior to enabling the peripheral clocks.
The clock for each peripheral is derived from the selected CPU subsystem. The clock mux controlled by this register is not glitch-free, therefore the CPUSELx register must be configured before the PCLKCRx register.
The reset for each peripheral is also driven from the selected CPU.

**Figure 3-127. CPUSEL1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | RESERVED | ECAP6 | ECAP5 | ECAP4 | ECAP3 | ECAP2 | ECAP1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-140. CPUSEL1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-8 | RESERVED | R-0 | 0h | Reserved |
| 7 | RESERVED | R/W | 0h | Reserved |
| 6 | RESERVED | R/W | 0h | Reserved |
| 5 | ECAP6 | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |
| 4 | ECAP5 | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |
| 3 | ECAP4 | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |
| 2 | ECAP3 | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |
| 1 | ECAP2 | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |
| 0 | ECAP1 | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.42 CPUSEL2 Register (Offset = DAh) [Reset = 00000000h]

CPUSEL2 is shown in Figure 3-128 and described in Table 3-141.

Return to the Summary Table.

CPU Select register for common peripherals
This register must be configured prior to enabling the peripheral clocks.
The clock for each peripheral is derived from the selected CPU subsystem. The clock mux controlled by this register is not glitch-free, therefore the CPUSELx register must be configured before the PCLKCRx register.
The reset for each peripheral is also driven from the selected CPU.

**Figure 3-128. CPUSEL2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | RESERVED | EQEP3 | EQEP2 | EQEP1 |
| R-0-0h | | | | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-141. CPUSEL2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-4 | RESERVED | R-0 | 0h | Reserved |
| 3 | RESERVED | R/W | 0h | Reserved |
| 2 | EQEP3 | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |
| 1 | EQEP2 | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |
| 0 | EQEP1 | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.43 CPUSEL3 Register (Offset = DCh) [Reset = 00000000h]

CPUSEL3 is shown in Figure 3-129 and described in Table 3-142.

Return to the Summary Table.

CPU Select register for common peripherals
This register must be configured prior to enabling the peripheral clocks.
The clock for each peripheral is derived from the selected CPU subsystem. The clock mux controlled by this register is not glitch-free, therefore the CPUSELx register must be configured before the PCLKCRx register.
The reset for each peripheral is also driven from the selected CPU.

**Figure 3-129. CPUSEL3 Register**

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

**Table 3-142. CPUSEL3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-8 | RESERVED | R-0 | 0h | Reserved |
| 7 | RESERVED | R/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 |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.44 CPUSEL4 Register (Offset = DEh) [Reset = 00000000h]

CPUSEL4 is shown in Figure 3-130 and described in Table 3-143.

Return to the Summary Table.

CPU Select register for common peripherals
This register must be configured prior to enabling the peripheral clocks.
The clock for each peripheral is derived from the selected CPU subsystem. The clock mux controlled by this register is not glitch-free, therefore the CPUSELx register must be configured before the PCLKCRx register.
The reset for each peripheral is also driven from the selected CPU.

**Figure 3-130. CPUSEL4 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | |
| R-0-0h | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | | | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | SD2 | SD1 |
| R-0-0h | | | | | | | | 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 3-143. CPUSEL4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-8 | RESERVED | R-0 | 0h | Reserved |
| 7 | RESERVED | R/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 | SD2 | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |
| 0 | SD1 | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.45 CPUSEL5 Register (Offset = E0h) [Reset = 00000000h]

CPUSEL5 is shown in Figure 3-131 and described in Table 3-144.

Return to the Summary Table.

CPU Select register for common peripherals
This register must be configured prior to enabling the peripheral clocks.
The clock for each peripheral is derived from the selected CPU subsystem. The clock mux controlled by this register is not glitch-free, therefore the CPUSELx register must be configured before the PCLKCRx register.
The reset for each peripheral is also driven from the selected CPU.

**Figure 3-131. CPUSEL5 Register**

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

**Table 3-144. CPUSEL5 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-4 | RESERVED | R-0 | 0h | Reserved |
| 3 | SCI_D | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |
| 2 | SCI_C | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |
| 1 | SCI_B | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |
| 0 | SCI_A | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.46 CPUSEL6 Register (Offset = E2h) [Reset = 00000000h]

CPUSEL6 is shown in Figure 3-132 and described in Table 3-145.

Return to the Summary Table.

CPU Select register for common peripherals
This register must be configured prior to enabling the peripheral clocks.
The clock for each peripheral is derived from the selected CPU subsystem. The clock mux controlled by this register is not glitch-free, therefore the CPUSELx register must be configured before the PCLKCRx register.
The reset for each peripheral is also driven from the selected CPU.

**Figure 3-132. CPUSEL6 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | RESERVED | RESERVED |
| R-0-0h | | | | | | R/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 | SPI_C | SPI_B | SPI_A |
| R-0-0h | | | | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-145. CPUSEL6 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 | SPI_C | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |
| 1 | SPI_B | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |
| 0 | SPI_A | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.47 CPUSEL7 Register (Offset = E4h) [Reset = 00000000h]

CPUSEL7 is shown in Figure 3-133 and described in Table 3-146.

Return to the Summary Table.

CPU Select register for common peripherals
This register must be configured prior to enabling the peripheral clocks.
The clock for each peripheral is derived from the selected CPU subsystem. The clock mux controlled by this register is not glitch-free, therefore the CPUSELx register must be configured before the PCLKCRx register.
The reset for each peripheral is also driven from the selected CPU.

**Figure 3-133. CPUSEL7 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | RESERVED | RESERVED |
| R-0-0h | | | | | | R/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 | | | | | | I2C_B | I2C_A |
| R-0-0h | | | | | | R/W-0h | R/W-0h |

**Table 3-146. CPUSEL7 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 |
| 15-2 | RESERVED | R-0 | 0h | Reserved |
| 1 | I2C_B | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |
| 0 | I2C_A | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.48 CPUSEL8 Register (Offset = E6h) [Reset = 00000000h]

CPUSEL8 is shown in Figure 3-134 and described in Table 3-147.

Return to the Summary Table.

CPU Select register for common peripherals
This register must be configured prior to enabling the peripheral clocks.
The clock for each peripheral is derived from the selected CPU subsystem. The clock mux controlled by this register is not glitch-free, therefore the CPUSELx register must be configured before the PCLKCRx register.
The reset for each peripheral is also driven from the selected CPU.

**Figure 3-134. CPUSEL8 Register**

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

**Table 3-147. CPUSEL8 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-4 | RESERVED | R-0 | 0h | Reserved |
| 3 | RESERVED | R/W | 0h | Reserved |
| 2 | RESERVED | R/W | 0h | Reserved |
| 1 | CAN_B | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |
| 0 | CAN_A | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.49 CPUSEL9 Register (Offset = E8h) [Reset = 00000000h]

CPUSEL9 is shown in Figure 3-135 and described in Table 3-148.

Return to the Summary Table.

CPU Select register for common peripherals
This register must be configured prior to enabling the peripheral clocks.
The clock for each peripheral is derived from the selected CPU subsystem. The clock mux controlled by this register is not glitch-free, therefore the CPUSELx register must be configured before the PCLKCRx register.
The reset for each peripheral is also driven from the selected CPU.

**Figure 3-135. CPUSEL9 Register**

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

**Table 3-148. CPUSEL9 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-2 | RESERVED | R-0 | 0h | Reserved |
| 1 | McBSP_B | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |
| 0 | McBSP_A | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.50 CPUSEL11 Register (Offset = ECh) [Reset = 00000000h]

CPUSEL11 is shown in Figure 3-136 and described in Table 3-149.

Return to the Summary Table.

CPU Select register for common peripherals
This register must be configured prior to enabling the peripheral clocks.
The clock for each peripheral is derived from the selected CPU subsystem. The clock mux controlled by this register is not glitch-free, therefore the CPUSELx register must be configured before the PCLKCRx register.
The reset for each peripheral is also driven from the selected CPU.

**Figure 3-136. CPUSEL11 Register**

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

**Table 3-149. CPUSEL11 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-4 | RESERVED | R-0 | 0h | Reserved |
| 3 | ADC_D | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Note:<br>[1] These CPUSEL bits affect the ownership of only ADC Configuration registers by CPU1 or CPU2 (which are mapped on the mapped to VBUS32). ADC result registers are readable from all masters without any CPUSEL dependency.<br>Reset type: CPU1.SYSRSn |
| 2 | ADC_C | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Note:<br>[1] These CPUSEL bits affect the ownership of only ADC Configuration registers by CPU1 or CPU2 (which are mapped on the mapped to VBUS32). ADC result registers are readable from all masters without any CPUSEL dependency.<br>Reset type: CPU1.SYSRSn |
| 1 | ADC_B | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Note:<br>[1] These CPUSEL bits affect the ownership of only ADC Configuration registers by CPU1 or CPU2 (which are mapped on the mapped to VBUS32). ADC result registers are readable from all masters without any CPUSEL dependency.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-149. CPUSEL11 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | ADC_A | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Note:<br>[1] These CPUSEL bits affect the ownership of only ADC Configuration registers by CPU1 or CPU2 (which are mapped on the mapped to VBUS32). ADC result registers are readable from all masters without any CPUSEL dependency.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.51 CPUSEL12 Register (Offset = EEh) [Reset = 00000000h]

CPUSEL12 is shown in Figure 3-137 and described in Table 3-150.

Return to the Summary Table.

CPU Select register for common peripherals
This register must be configured prior to enabling the peripheral clocks.
The clock for each peripheral is derived from the selected CPU subsystem. The clock mux controlled by this register is not glitch-free, therefore the CPUSELx register must be configured before the PCLKCRx register.
The reset for each peripheral is also driven from the selected CPU.

**Figure 3-137. CPUSEL12 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CMPSS8 | CMPSS7 | CMPSS6 | CMPSS5 | CMPSS4 | CMPSS3 | CMPSS2 | CMPSS1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-150. CPUSEL12 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-8 | RESERVED | R-0 | 0h | Reserved |
| 7 | CMPSS8 | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |
| 6 | CMPSS7 | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |
| 5 | CMPSS6 | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |
| 4 | CMPSS5 | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |
| 3 | CMPSS4 | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |
| 2 | CMPSS3 | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |
| 1 | CMPSS2 | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-150. CPUSEL12 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | CMPSS1 | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.52 CPUSEL14 Register (Offset = F2h) [Reset = 00000000h]

CPUSEL14 is shown in Figure 3-138 and described in Table 3-151.

Return to the Summary Table.

CPU Select register for common peripherals
This register must be configured prior to enabling the peripheral clocks.
The clock for each peripheral is derived from the selected CPU subsystem. The clock mux controlled by this register is not glitch-free, therefore the CPUSELx register must be configured before the PCLKCRx register.
The reset for each peripheral is also driven from the selected CPU.

**Figure 3-138. CPUSEL14 Register**

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

**Table 3-151. CPUSEL14 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-20 | RESERVED | R-0 | 0h | Reserved |
| 19 | RESERVED | R/W | 0h | Reserved |
| 18 | DAC_C | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |
| 17 | DAC_B | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |
| 16 | DAC_A | R/W | 0h | 0: Connected to CPU1<br>1: Connected to CPU2<br>Reset type: CPU1.SYSRSn |
| 15-4 | RESERVED | R-0 | 0h | Reserved |
| 3 | RESERVED | R/W | 0h | Reserved |
| 2 | RESERVED | R/W | 0h | Reserved |
| 1 | RESERVED | R/W | 0h | Reserved |
| 0 | RESERVED | R/W | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.53 CPU2RESCTL Register (Offset = 122h) [Reset = 00000001h]

CPU2RESCTL is shown in Figure 3-139 and described in Table 3-152.

Return to the Summary Table.

CPU2 Reset Control Register

**Figure 3-139. CPU2RESCTL Register**

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

**Table 3-152. CPU2RESCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | KEY | R-0/W | 0h | Write to this register succeeds only if this field is written with a value of 0xa5a5<br>Note:<br>[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<br>Reset type: N/A |
| 15-1 | RESERVED | R-0 | 0h | Reserved |
| 0 | RESET | R/W | 1h | This bit controls the reset input of CPU2 core.<br>1: CPU2 is held in reset (CPU2.RSn = 0)<br>0: CPU2 reset is deactivated (CPU2.RSn = 1)<br>Note:<br>[1] If CPU2 is not used at-all by an application, it's advisable to put CPU2 in STANDBY mode rather than in reset to save on active power component on the CPU2 subsystem. This is because, all clocks keep toggling when reset is active on the CPU2 sub-system.<br>[2] Note: If CPU2 is in Standby mode, writing to this bit will have no effect. CPU2 may be reset by any Chip-level reset (POR, XRSn, CPU1.WDRSn, or CPU1.NMIWDRSn) or HIBRESETn. Alternately CPU2 may be woken up by any configured wake-up event.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.54 RSTSTAT Register (Offset = 124h) [Reset = 0000h]

RSTSTAT is shown in Figure 3-140 and described in Table 3-153.

Return to the Summary Table.

Reset Status register for secondary C28x CPUs

**Figure 3-140. RSTSTAT Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | CPU2HWBISTRST1 | CPU2HWBISTRST0 | CPU2NMIWDRST | CPU2RES |
| R-0-0h | | | | R/W1S-0h | R/W1S-0h | R/W1S-0h | R-0h |

**Table 3-153. RSTSTAT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-4 | RESERVED | R-0 | 0h | Reserved |
| 3 | CPU2HWBISTRST1 | R/W1S | 0h | CPU2HWBISTRST0 and CPU2HWBISTRST1 together indicates whether a HWBIST reset was issued to CPU2 or not<br>00: CPU2 was not reset by the CPU2 HWBIST<br>11: CPU2 was reset due to CPU2 HWBIST reset<br>This status bit is a latched flag. This flag can be cleared by the CPU1 by writing a 1<br>Reset type: CPU1.SYSRSn |
| 2 | CPU2HWBISTRST0 | R/W1S | 0h | CPU2HWBISTRST0 and CPU2HWBISTRST1 together indicates whether a HWBIST reset was issued to CPU2 or not<br>00: CPU2 was not reset by the CPU2 HWBIST<br>11: CPU2 was reset due to CPU2 HWBIST reset<br>This status bit is a latched flag. This flag can be cleared by the CPU1 by writing a 1<br>Reset type: CPU1.SYSRSn |
| 1 | CPU2NMIWDRST | R/W1S | 0h | Indicates whether a CPU2.NMIWD reset was issued to CPU2 or not<br>0: CPU2 was not reset by the CPU2.NMIWD<br>1: CPU2 was reset due to CPU2.NMIWD reset<br>This status bit is a latched flag.This flag can be cleared by the CPU1 by writing a 1<br>Reset type: CPU1.SYSRSn |
| 0 | CPU2RES | R | 0h | Reset status of CPU2 to CPU1<br>0: CPU2 core is in reset<br>1: CPU2 core is out of reset<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.55 LPMSTAT Register (Offset = 125h) [Reset = 0000h]

LPMSTAT is shown in Figure 3-141 and described in Table 3-154.

Return to the Summary Table.

LPM Status Register for secondary C28x CPUs

**Figure 3-141. LPMSTAT Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | CPU2LPMSTAT | |
| R-0-0h | | | | | | R-0h | |

**Table 3-154. LPMSTAT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-2 | RESERVED | R-0 | 0h | Reserved |
| 1-0 | CPU2LPMSTAT | R | 0h | These bits indicate the power mode CPU2<br>00: CPU2 is in ACTIVE mode<br>01: CPU2 is in IDLE mode<br>10: CPU2 is in STANDBY mode<br>11: Reserved<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.10.56 SYSDBGCTL Register (Offset = 12Ch) [Reset = 00000000h]

SYSDBGCTL is shown in Figure 3-142 and described in Table 3-155.

Return to the Summary Table.

System Debug Control register

**Figure 3-142. SYSDBGCTL 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 | | | | | | | BIT_0 |
| R-0-0h | | | | | | | R/W-0h |

**Table 3-155. SYSDBGCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-1 | RESERVED | R-0 | 0h | Reserved |
| 0 | BIT_0 | R/W | 0h | This bit is for use in PLL startup and is only reset by POR.<br>Reset type: POR |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.11 CLK_CFG_REGS Registers

Table 3-156 lists the memory-mapped registers for the CLK_CFG_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. CLK_CFG_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | CLKSEM | Clock Control Semaphore Register | EALLOW | Go |
| 2h | CLKCFGLOCK1 | Lock bit for CLKCFG registers | EALLOW | Go |
| 8h | CLKSRCCTL1 | Clock Source Control register-1 | EALLOW | Go |
| Ah | CLKSRCCTL2 | Clock Source Control register-2 | EALLOW | Go |
| Ch | CLKSRCCTL3 | Clock Source Control register-3 | EALLOW | Go |
| Eh | SYSPLLCTL1 | SYSPLL Control register-1 | EALLOW | Go |
| 14h | SYSPLLMULT | SYSPLL Multiplier register | EALLOW | Go |
| 16h | SYSPLLSTS | SYSPLL Status register | | Go |
| 18h | AUXPLLCTL1 | AUXPLL Control register-1 | EALLOW | Go |
| 1Eh | AUXPLLMULT | AUXPLL Multiplier register | EALLOW | Go |
| 20h | AUXPLLSTS | AUXPLL Status register | | Go |
| 22h | SYSCLKDIVSEL | System Clock Divider Select register | EALLOW | Go |
| 24h | AUXCLKDIVSEL | Auxillary Clock Divider Select register | EALLOW | Go |
| 26h | PERCLKDIVSEL | Peripheral Clock Divider Selet register | EALLOW | Go |
| 28h | XCLKOUTDIVSEL | XCLKOUT Divider Select register | EALLOW | Go |
| 2Ch | LOSPCP | Low Speed Clock Source Prescalar | EALLOW | Go |
| 2Eh | MCDCR | Missing Clock Detect Control Register | EALLOW | Go |
| 30h | X1CNT | 10-bit Counter on X1 Clock | | Go |

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. CLK_CFG_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| R-0 | R<br>-0 | Read<br>Returns 0s |
| Write Type | | |
| W | W | Write |
| W1S | W<br>1S | Write<br>1 to set |
| WSonce | W<br>Sonce | Write<br>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. |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-157. 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. |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.11.1 CLKSEM Register (Offset = 0h) [Reset = 00000000h]

CLKSEM is shown in Figure 3-143 and described in Table 3-158.

Return to the Summary Table.

Clock Control Semaphore Register

**Figure 3-143. CLKSEM Register**

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

**Table 3-158. CLKSEM Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | KEY | R-0/W | 0h | Writing the value 0xa5a5 will allow the writing of the SEM bits, else writes are ignored. Reads will return 0.<br>Note:<br>[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<br>Reset type: N/A |
| 15-2 | RESERVED | R-0 | 0h | Reserved |
| 1-0 | SEM | R/W | 0h | This register provides a mechanism to acquire all the CLKCFG registers (except this register) by CPU1 or CPU2. A CPU can perform read/writes to any of the CLKCFG registers (except this register) only if it owns the semaphore. Otherwise, writes are ignored and reads will return 0x0.<br>Semaphore State Transitions:<br>A value of 00, 10, 11 gives ownership to CPU1<br>A value of 01 gives ownership to CPU2.<br>The following are the only state transitions allowed on these bits.<br>00,11 <-> 01 (allowed by CPU2)<br>00,11 <-> 10 (allowed by CPU1)<br>If a CPU doesn't own the CLK_CFG_REGS set of registers (as defined by the state of this semaphore), reads from that CPU to all those registers return 0x0 and writes are ignore. Note that this is not true of CLKSEM register. The CLKSEM register's reads and writes are always allowed from both CPU1 and CPU2.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.11.2 CLKCFGLOCK1 Register (Offset = 2h) [Reset = 00000000h]

CLKCFGLOCK1 is shown in Figure 3-144 and described in Table 3-159.

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 3-144. CLKCFGLOCK1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| LOSPCP | RESERVED | PERCLKDIVSEL | AUXCLKDIVSEL | SYSCLKDIVSEL | AUXPLLMULT | RESERVED | RESERVED |
| R/WSonce-0h | R-0-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R-0-0h | R-0-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| AUXPLLCTL1 | SYSPLLMULT | SYSPLLCTL3 | SYSPLLCTL2 | SYSPLLCTL1 | CLKSRCCTL3 | CLKSRCCTL2 | CLKSRCCTL1 |
| R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h |

**Table 3-159. CLKCFGLOCK1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15 | LOSPCP | R/WSonce | 0h | Lock bit for LOSPCP register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: CPU1.SYSRSn |
| 14 | RESERVED | R-0 | 0h | Reserved |
| 13 | PERCLKDIVSEL | R/WSonce | 0h | Lock bit for PERCLKDIVSEL register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: CPU1.SYSRSn |
| 12 | AUXCLKDIVSEL | R/WSonce | 0h | Lock bit for AUXCLKDIVSEL register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: CPU1.SYSRSn |
| 11 | SYSCLKDIVSEL | R/WSonce | 0h | Lock bit for SYSCLKDIVSEL register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: CPU1.SYSRSn |
| 10 | AUXPLLMULT | R/WSonce | 0h | Lock bit for AUXPLLMULT register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: CPU1.SYSRSn |
| 9 | RESERVED | R-0 | 0h | Reserved |
| 8 | RESERVED | R-0 | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-159. CLKCFGLOCK1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 7 | AUXPLLCTL1 | R/WSonce | 0h | Lock bit for AUXPLLCTL1 register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: CPU1.SYSRSn |
| 6 | SYSPLLMULT | R/WSonce | 0h | Lock bit for SYSPLLMULT register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: CPU1.SYSRSn |
| 5 | SYSPLLCTL3 | R/WSonce | 0h | Lock bit for SYSPLLCTL3 register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: CPU1.SYSRSn |
| 4 | SYSPLLCTL2 | R/WSonce | 0h | Lock bit for SYSPLLCTL2 register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: CPU1.SYSRSn |
| 3 | SYSPLLCTL1 | R/WSonce | 0h | Lock bit for SYSPLLCTL1 register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: CPU1.SYSRSn |
| 2 | CLKSRCCTL3 | R/WSonce | 0h | Lock bit for CLKSRCCTL3 register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: CPU1.SYSRSn |
| 1 | CLKSRCCTL2 | R/WSonce | 0h | Lock bit for CLKSRCCTL2 register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: CPU1.SYSRSn |
| 0 | CLKSRCCTL1 | R/WSonce | 0h | Lock bit for CLKSRCCTL1 register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.11.3 CLKSRCCTL1 Register (Offset = 8h) [Reset = 00000000h]

CLKSRCCTL1 is shown in Figure 3-145 and described in Table 3-160.

Return to the Summary Table.

Clock Source Control register-1
This memory mapped register requires a delay of 69 SYSCLK cycles between subsequent writes to the register, otherwise a second write can be lost. This delay can be realized by adding 69 NOP instructions.

**Figure 3-145. CLKSRCCTL1 Register**

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

**Table 3-160. CLKSRCCTL1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-6 | RESERVED | R-0 | 0h | Reserved |
| 5 | WDHALTI | R/W | 0h | Watchdog HALT Mode Ignore Bit: This bit determines if CPU1.WD is functional in the HALT mode or not.<br>0 = CPU1.WD is not functional in the HALT mode. Clock to CPU1.WD is gated when system enters HALT mode. Additionally, INTOSC1 and INTOSC2 are powered-down when system enters HALT mode<br>1 = CPU1.WD is functional in the HALT mode. Clock to CPU1.WD is not gated and INTOSC1/2 are not powered-down when system enters HALT mode<br>Notes:<br>[1] Clock to CPU2.WD clocks is always gated in the HALT mode.<br>Reset type: XRSn |
| 4 | XTALOFF | R/W | 0h | Crystal (External) Oscillator Off Bit: This bit turns external oscillator off:<br>0 = Crystal (External) Oscillator On (default on reset)<br>1 = Crystal (External) Oscillator Off<br>NOTE: Ensure no resources are using a clock source prior to disabling it. For example OSCCLKSRCSEL (SYSPLL), AUXOSCCLKSRCSEL (AUXPLL), CANxBCLKSEL (CAN Clock), TMR2CLKSRCSEL (CPUTIMER2) and XCLKOUTSEL(XCLKOUT).<br>Reset type: XRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-160. CLKSRCCTL1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3 | INTOSC2OFF | R/W | 0h | Internal Oscillator 2 Off Bit: This bit turns oscillator 2 off:<br>0 = Internal Oscillator 2 On (default on reset)<br>1 = Internal Oscillator 2 Off<br>This bit could be used by the user to turn off the internal oscillator 2 if it is not used.<br>NOTE: Ensure no resources are using a clock source prior to disabling it. For example OSCCLKSRCSEL (SYSPLL), AUXOSCCLKSRCSEL (AUXPLL), TMR2CLKSRCSEL (CPUTIMER2) and XCLOCKOUT (XCLKOUT).<br>Reset type: XRSn |
| 2 | RESERVED | R-0 | 0h | Reserved |
| 1-0 | OSCCLKSRCSEL | R/W | 0h | Oscillator Clock Source Select Bit: This bit selects the source for OSCCLK.<br>00 = INTOSC2 (default on reset)<br>01 = External Oscillator (XTAL)<br>10 = INTOSC1<br>11 = reserved (default to INTOSC1)<br>At power-up or after an XRSn, INTOSC2 is selected by default. Whenever the user changes the clock source using these bits, the SYSPLLMULT register will be forced to zero and the PLL will be bypassed and powered down. This prevents potential PLL overshoot. The user will then have to write to the SYSPLLMULT register to configure the appropriate multiplier.<br>The user must wait 10 OSCCLK cycles before writing to SYSPLLMULT<br>or disabling the previous clock source to allow the change to complete..<br>Notes:<br>[1] Reserved selection defaults to 00 configuration<br>[2] INTOSC1 is recommended to be used only after missing clock detection. If user wants to re-lock the PLL with INTOSC1 (the back-up clock source) after missing clock is detected, he can do a MCLKCLR and lock the PLL.<br>Reset type: XRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.11.4 CLKSRCCTL2 Register (Offset = Ah) [Reset = 00000000h]

CLKSRCCTL2 is shown in Figure 3-146 and described in Table 3-161.

Return to the Summary Table.

Clock Source Control register-2
This memory mapped register requires a delay of 69 SYSCLK cycles between subsequent writes to the register, otherwise a second write can be lost. This delay can be realized by adding 69 NOP instructions.

**Figure 3-146. CLKSRCCTL2 Register**

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

**Table 3-161. CLKSRCCTL2 Register Field Descriptions**

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



Copyright © 2024 Texas Instruments Incorporated

**Table 3-161. CLKSRCCTL2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1-0 | AUXOSCCLKSRCSEL | R/W | 0h | Oscillator Clock Source Select Bit: This bit selects the source for AUXOSCCLK:<br>00 = INTOSC2 (default on reset)<br>01 = External Oscillator (XTAL)<br>10 = AUXCLKIN (from GPIO)<br>11 = Reserved<br>Whenever the user changes the clock source using these bits, the AUXPLLMULT register will be forced to zero and the PLL will be bypassed and powered down. This prevents potential PLL overshoot. The user will then have to write to the AUXPLLMULT register to configure the appropriate multiplier.<br>The user must wait 10 OSCCLK cycles before writing to AUXPLLMULT<br>or disabling the previous clock source to allow the change to complete.<br>The missing clock detection circuit does not affect these bits.<br>Reset type: XRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.11.5 CLKSRCCTL3 Register (Offset = Ch) [Reset = 00000000h]

CLKSRCCTL3 is shown in Figure 3-147 and described in Table 3-162.

Return to the Summary Table.

Clock Source Control register-3
This memory mapped register requires a delay of 69 SYSCLK cycles between subsequent writes to the register, otherwise a second write can be lost. This delay can be realized by adding 69 NOP instructions.

**Figure 3-147. CLKSRCCTL3 Register**

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

**Table 3-162. CLKSRCCTL3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-3 | RESERVED | R-0 | 0h | Reserved |
| 2-0 | XCLKOUTSEL | R/W | 0h | XCLKOUT Source Select Bit: This bit selects the source for XCLKOUT:<br>000 = PLLSYSCLK (default on reset)<br>001 = PLLRAWCLK<br>010 = CPU1.SYSCLK<br>011 = CPU2.SYSCLK<br>100 = AUXPLLRAWCLK<br>101 = INTOSC1<br>110 = INTOSC2<br>111 = Reserved<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.11.6 SYSPLLCTL1 Register (Offset = Eh) [Reset = 00000000h]

SYSPLLCTL1 is shown in Figure 3-148 and described in Table 3-163.

Return to the Summary Table.

SYSPLL Control register-1
This memory mapped register requires a delay of 69 SYSCLK cycles between subsequent writes to the register, otherwise a second write can be lost. This delay can be realized by adding 69 NOP instructions.

**Figure 3-148. SYSPLLCTL1 Register**

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

**Table 3-163. SYSPLLCTL1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-2 | RESERVED | R-0 | 0h | Reserved |
| 1 | PLLCLKEN | R/W | 0h | SYSPLL bypassed or included in the PLLSYSCLK path: This bit decides if the SYSPLL is bypassed when PLLSYSCLK is generated<br>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.<br>0 = SYSPLL is bypassed. Clock to system is direct feed from OSCCLK<br>Reset type: XRSn |
| 0 | PLLEN | R/W | 0h | SYSPLL enabled or disabled: This bit decides if the SYSPLL is enabled or not<br>1 = SYSPLL is enabled<br>0 = SYSPLL is powered off. Clock to system is direct feed from OSCCLK<br>Reset type: XRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.11.7 SYSPLLMULT Register (Offset = 14h) [Reset = 00000000h]

SYSPLLMULT is shown in Figure 3-149 and described in Table 3-164.

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 69 SYSCLK cycles between subsequent writes to the register, otherwise a second write can be lost. This delay can be realized by adding 69 NOP instructions.

**Figure 3-149. SYSPLLMULT Register**

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

**Table 3-164. SYSPLLMULT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-10 | RESERVED | R-0 | 0h | Reserved |
| 9-8 | FMULT | R/W | 0h | SYSPLL Fractional Multiplier:<br>00 Fractional Multiplier = 0<br>01 Fractional Multiplier = 0.25<br>10 Fractional Multiplier = 0.5<br>11 Fractional Multiplier = 0.75<br>Reset type: XRSn |
| 7 | RESERVED | R-0 | 0h | Reserved |
| 6-0 | IMULT | R/W | 0h | SYSPLL Integer Multiplier:<br>For 0000000 Fout = Fref (PLLBYPASS) Integer Multiplier = 1<br>0000001 Integer Multiplier = 1<br>0000010 Integer Multiplier = 2<br>0000011 Integer Multiplier = 3<br>.....<br>1111111 Integer Multipler = 127<br>Reset type: XRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.11.8 SYSPLLSTS Register (Offset = 16h) [Reset = 00000000h]

SYSPLLSTS is shown in Figure 3-150 and described in Table 3-165.

Return to the Summary Table.

SYSPLL Status register

**Figure 3-150. SYSPLLSTS Register**

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

**Table 3-165. SYSPLLSTS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-2 | RESERVED | R-0 | 0h | Reserved |
| 1 | SLIPS | R | 0h | SYSPLL Slip Status Bit: This bit indicates whether the SYSPLL is out of lock range<br>0 = SYSPLL is not out of lock<br>1 = SYSPLL is out of loc<br>The SLIPS bit will only be set on a PLL slip condition after the PLL is used as the SYSCLK source by seting the SYSPLLCTL1[PLLCLKEN] bit. Disabling and re-enabling the PLL with PLLEN is the only way to clear this bit.<br>Note:<br>[1] If SYSPLL out of lock condition is detected then interrupts are fired to CPU1 and CPU2 through their respective ePIE modules. Software can decide to relock the PLL or switch to PLL bypass mode in the interrupt handler<br>Reset type: XRSn |
| 0 | LOCKS | R | 0h | SYSPLL Lock Status Bit: This bit indicates whether the SYSPLL is locked or not<br>0 = SYSPLL is not yet locked<br>1 = SYSPLL is locked<br>Reset type: XRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.11.9 AUXPLLCTL1 Register (Offset = 18h) [Reset = 00000000h]

AUXPLLCTL1 is shown in Figure 3-151 and described in Table 3-166.

Return to the Summary Table.

AUXPLL Control register-1

**Figure 3-151. AUXPLLCTL1 Register**

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

**Table 3-166. AUXPLLCTL1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-2 | RESERVED | R-0 | 0h | Reserved |
| 1 | PLLCLKEN | R/W | 0h | AUXPLL bypassed or included in the AUXPLLCLK path: This bit decides if the AUXPLL is bypassed when AUXPLLCLK is generated<br>1 = AUXPLLCLK is fed from the AUXPLL clock output. Users need to make sure that the PLL is locked before enabling this clock to the AUXPLLCLK connected modules.<br>0 = AUXPLL is bypassed. Clock to modules connected to AUXPLLCLK is direct feed from AUXOSCCLK<br>Reset type: XRSn |
| 0 | PLLEN | R/W | 0h | AUXPLL enabled or disabled: This bit decides if the AUXPLL is enabled or not<br>1 = AUXPLL is enabled<br>0 = AUXPLL is powered off. Clock to system is direct feed from AUXOSCCLK<br>Reset type: XRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.11.10 AUXPLLMULT Register (Offset = 1Eh) [Reset = 00000000h]

AUXPLLMULT is shown in Figure 3-152 and described in Table 3-167.

Return to the Summary Table.

AUXPLL 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 69 SYSCLK cycles between subsequent writes to the register, otherwise a second write can be lost. This delay can be realized by adding 69 NOP instructions.

**Figure 3-152. AUXPLLMULT Register**

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

**Table 3-167. AUXPLLMULT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-10 | RESERVED | R-0 | 0h | Reserved |
| 9-8 | FMULT | R/W | 0h | AUXPLL Fractional Multiplier :<br>00 Fractional Multiplier = 0<br>01 Fractional Multiplier = 0.25<br>10 Fractional Multiplier = 0.5<br>11 Fractional Multiplier = 0.75<br>Reset type: XRSn |
| 7 | RESERVED | R-0 | 0h | Reserved |
| 6-0 | IMULT | R/W | 0h | AUXPLL Integer Multiplier:<br>For 0000000 Fout = Fref (PLLBYPASS) Integer Multiplier = 1<br>0000001 Integer Multiplier = 1<br>0000010 Integer Multiplier = 2<br>0000011 Integer Multiplier = 3<br>.....<br>1111111 Integer Multipler = 127<br>Reset type: XRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.11.11 AUXPLLSTS Register (Offset = 20h) [Reset = 00000000h]

AUXPLLSTS is shown in Figure 3-153 and described in Table 3-168.

Return to the Summary Table.

AUXPLL Status register

**Figure 3-153. AUXPLLSTS Register**

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

**Table 3-168. AUXPLLSTS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-2 | RESERVED | R-0 | 0h | Reserved |
| 1 | SLIPS | R | 0h | AUXPLL Slip Status Bit: This bit indicates whether the AUXPLL is out of lock range<br>0 = AUXPLL is not out of lock<br>1 = AUXPLL is out of lock<br>The SLIPS bit will only be set on a PLL slip condition after the PLL is used as the AUXPLLCLK source by seting the AUXPLLCTL1[PLLCLKEN] bit. Disabling and re-enabling the PLL with PLLEN is the only way to clear this bit.<br>Note:<br>[1] If AUXPLL out of lock condition is detected then interrupts are fired to CPU1 and CPU2 through their respective ePIE modules. Software can decide to relock the PLL or switch to PLL bypass mode in the interrupt handler<br>Reset type: XRSn |
| 0 | LOCKS | R | 0h | AUXPLL Lock Status Bit: This bit indicates whether the AUXPLL is locked or not<br>0 = AUXPLL is not yet locked<br>1 = AUXPLL is locked<br>Reset type: XRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.11.12 SYSCLKDIVSEL Register (Offset = 22h) [Reset = 00000002h]

SYSCLKDIVSEL is shown in Figure 3-154 and described in Table 3-169.

Return to the Summary Table.

System Clock Divider Select register
This memory mapped register requires a delay of 69 SYSCLK cycles between subsequent writes to the register, otherwise a second write can be lost. This delay can be realized by adding 69 NOP instructions.

**Figure 3-154. SYSCLKDIVSEL Register**

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

**Table 3-169. SYSCLKDIVSEL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-6 | RESERVED | R-0 | 0h | Reserved |
| 5-0 | PLLSYSCLKDIV | R/W | 2h | PLLSYSCLK Divide Select: This bit selects the divider setting for the PLLSYSCLK.<br>000000 = /1<br>000001 = /2<br>000010 = /4 (default on reset)<br>000011 = /6<br>000100 = /8<br>.....<br>111111 = /126<br>Reset type: XRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.11.13 AUXCLKDIVSEL Register (Offset = 24h) [Reset = 00000001h]

AUXCLKDIVSEL is shown in Figure 3-155 and described in Table 3-170.

Return to the Summary Table.

Auxillary Clock Divider Select register
This memory mapped register requires a delay of 69 SYSCLK cycles between subsequent writes to the register, otherwise a second write can be lost. This delay can be realized by adding 69 NOP instructions.

**Figure 3-155. AUXCLKDIVSEL Register**

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

**Table 3-170. AUXCLKDIVSEL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-2 | RESERVED | R-0 | 0h | Reserved |
| 1-0 | AUXPLLDIV | R/W | 1h | AUXPLLCLK Divide Select: This bit selects the divider setting for the AUXPLLCK.<br>00 = /1<br>01 = /2 (default on reset)<br>10 = /4<br>11 = /8<br>Reset type: XRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.11.14 PERCLKDIVSEL Register (Offset = 26h) [Reset = 00000051h]

PERCLKDIVSEL is shown in Figure 3-156 and described in Table 3-171.

Return to the Summary Table.

Peripheral Clock Divider Selet register
This memory mapped register requires a delay of 69 SYSCLK cycles between subsequent writes to the register, otherwise a second write can be lost. This delay can be realized by adding 69 NOP instructions.

**Figure 3-156. PERCLKDIVSEL Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | EMIF2CLKDIV | RESERVED | EMIF1CLKDIV | RESERVED | | EPWMCLKDIV | |
| R-0-0h | R/W-1h | R-0-0h | R/W-1h | R/W-0h | | R/W-1h | |

**Table 3-171. PERCLKDIVSEL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-7 | RESERVED | R-0 | 0h | Reserved |
| 6 | EMIF2CLKDIV | R/W | 1h | EMIF2 Clock Divide Select: This bit selects whether the EMIF2 module run with a /1 or /2 clock.<br>0: /1 of CPU1.SYSCLK is selected<br>1: /2 of CPU1.SYSCLK is selected<br>Reset type: CPU1.SYSRSn |
| 5 | RESERVED | R-0 | 0h | Reserved |
| 4 | EMIF1CLKDIV | R/W | 1h | EMIF1 Clock Divide Select: This bit selects whether the EMIF1 module run with a /1 or /2 clock.<br>For single core device<br>0: /1 of CPU1.SYSCLK is selected<br>1: /2 of CPU1.SYSCLK is selected<br>For Dual core device<br>0: /1 of PLLSYSCLK is selected<br>1: /2 of PLLSYSCLK is selected<br>Reset type: CPU1.SYSRSn |
| 3-2 | RESERVED | R/W | 0h | Reserved |
| 1-0 | EPWMCLKDIV | R/W | 1h | EPWM Clock Divide Select: This bit selects whether the EPWM modules run with a /1 or /2 clock. This divider sits in front of the PLLSYSCLK<br>x0 = /1 of PLLSYSCLK<br>x1 = /2 of PLLSYSLCK (default on reset)<br>Note: /1 should only be used when SYSCLK is 100MHz or less, see the datasheet for EPWMCLK specifications<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.11.15 XCLKOUTDIVSEL Register (Offset = 28h) [Reset = 00000003h]

XCLKOUTDIVSEL is shown in Figure 3-157 and described in Table 3-172.

Return to the Summary Table.

XCLKOUT Divider Select register
This memory mapped register requires a delay of 69 SYSCLK cycles between subsequent writes to the register, otherwise a second write can be lost. This delay can be realized by adding 69 NOP instructions.

**Figure 3-157. 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 3-172. XCLKOUTDIVSEL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-2 | RESERVED | R-0 | 0h | Reserved |
| 1-0 | XCLKOUTDIV | R/W | 3h | XCLKOUT Divide Select: This bit selects the divider setting for the XCLKOUT.<br>00 = /1<br>01 = /2<br>10 = /4<br>11 = /8 (default on reset)<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.11.16 LOSPCP Register (Offset = 2Ch) [Reset = 00000002h]

LOSPCP is shown in Figure 3-158 and described in Table 3-173.

Return to the Summary Table.

Low Speed Clock Source Prescalar

**Figure 3-158. 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 3-173. LOSPCP Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-3 | RESERVED | R-0 | 0h | Reserved |
| 2-0 | LSPCLKDIV | R/W | 2h | These bits configure the low-speed peripheral clock (LSPCLK) rate relative to SYSCLK of CPU1 and CPU2.<br>000,LSPCLK = / 1<br>001,LSPCLK = / 2<br>010,LSPCLK = / 4 (default on reset)<br>011,LSPCLK = / 6<br>100,LSPCLK = / 8<br>101,LSPCLK = / 10<br>110,LSPCLK = / 12<br>111,LSPCLK = / 14<br>Note:<br>[1] This clock is used as strobe for the SCI and SPI modules.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.11.17 MCDCR Register (Offset = 2Eh) [Reset = 00000000h]

MCDCR is shown in Figure 3-159 and described in Table 3-174.

Return to the Summary Table.

Missing Clock Detect Control Register

**Figure 3-159. MCDCR Register**

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

**Table 3-174. MCDCR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-4 | RESERVED | R-0 | 0h | Reserved |
| 3 | OSCOFF | R/W | 0h | Oscillator Clock Off Bit:<br>0 = OSCCLK Connected to OSCCLK Counter in MCD module<br>1 = OSCCLK Disconnected to OSCCLK Counter in MCD module<br>Reset type: XRSn |
| 2 | MCLKOFF | R/W | 0h | Missing Clock Detect Off Bit:<br>0 = Missing Clock Detect Circuit Enabled<br>1 = Missing Clock Detect Circuit Disabled<br>Reset type: XRSn |
| 1 | MCLKCLR | R-0/W1S | 0h | Missing Clock Clear Bit:<br>Write 1' to this bit to clear MCLKSTS bit and reset the missing clock detect circuit.'<br>Reset type: XRSn |
| 0 | MCLKSTS | R | 0h | Missing Clock Status Bit:<br>0 = OSCCLK Is OK<br>1 = OSCCLK Detected Missing, CLOCKFAILn Generated<br>Reset type: XRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.11.18 X1CNT Register (Offset = 30h) [Reset = 00000000h]

X1CNT is shown in Figure 3-160 and described in Table 3-175.

Return to the Summary Table.

10-bit Counter on X1 Clock

**Figure 3-160. X1CNT Register**

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

**Table 3-175. X1CNT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-10 | RESERVED | R-0 | 0h | Reserved |
| 9-0 | X1CNT | R | 0h | X1 Counter:<br>- This counter increments on every X1 CLOCKs positive-edge.<br>- Once it reaches the values of 0x3ff, it freezes<br>- 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. Note: Since this bit counter cannot be reset locally, TI recommends using the 'SysCtl_pollCpuTimer' function in C2000Ware to detect the validity of the clock on X1<br>Reset type: POR |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.12 CPU_SYS_REGS Registers

Table 3-176 lists the memory-mapped registers for the CPU_SYS_REGS registers. All register offset addresses not listed in Table 3-176 should be considered as reserved locations and the register contents should not be modified.

**Table 3-176. CPU_SYS_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | CPUSYSLOCK1 | Lock bit for CPUSYS registers | EALLOW | Go |
| 6h | HIBBOOTMODE | HIB Boot Mode Register | EALLOW | Go |
| 8h | IORESTOREADDR | IORestore() routine Address Register | EALLOW | Go |
| Ah | PIEVERRADDR | PIE Vector Fetch Error Address register | EALLOW | Go |
| 22h | PCLKCR0 | Peripheral Clock Gating Registers | EALLOW | Go |
| 24h | PCLKCR1 | Peripheral Clock Gating Registers | EALLOW | Go |
| 26h | PCLKCR2 | Peripheral Clock Gating Registers | EALLOW | Go |
| 28h | PCLKCR3 | Peripheral Clock Gating Registers | EALLOW | Go |
| 2Ah | PCLKCR4 | Peripheral Clock Gating Registers | EALLOW | Go |
| 2Eh | PCLKCR6 | Peripheral Clock Gating Registers | EALLOW | Go |
| 30h | PCLKCR7 | Peripheral Clock Gating Registers | EALLOW | Go |
| 32h | PCLKCR8 | Peripheral Clock Gating Registers | EALLOW | Go |
| 34h | PCLKCR9 | Peripheral Clock Gating Registers | EALLOW | Go |
| 36h | PCLKCR10 | Peripheral Clock Gating Registers | EALLOW | Go |
| 38h | PCLKCR11 | Peripheral Clock Gating Registers | EALLOW | Go |
| 3Ah | PCLKCR12 | Peripheral Clock Gating Registers | EALLOW | Go |
| 3Ch | PCLKCR13 | Peripheral Clock Gating Registers | EALLOW | Go |
| 3Eh | PCLKCR14 | Peripheral Clock Gating Registers | EALLOW | Go |
| 42h | PCLKCR16 | Peripheral Clock Gating Registers | EALLOW | Go |
| 74h | SECMSEL | Secondary Master Select register for common peripherals: Selects between CLA & DMA | EALLOW | Go |
| 76h | LPMCR | LPM Control Register | EALLOW | Go |
| 78h | GPIOLPMSEL0 | GPIO LPM Wakeup select registers | EALLOW | Go |
| 7Ah | GPIOLPMSEL1 | GPIO LPM Wakeup select registers | EALLOW | Go |
| 7Ch | TMR2CLKCTL | Timer2 Clock Measurement functionality control register | EALLOW | Go |
| 80h | RESC | Reset Cause register | | Go |

Complex bit access types are encoded to fit into small table cells. Table 3-177 shows the codes that are used for access types in this section.

**Table 3-177. CPU_SYS_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| R-0 | R<br>-0 | Read<br>Returns 0s |
| Write Type | | |
| W | W | Write |
| W1C | W<br>1C | Write<br>1 to clear |
| W1S | W<br>1S | Write<br>1 to set |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-177. CPU_SYS_REGS Access Type Codes (continued)**

| Access Type | Code | Description |
|---|---|---|
| WSonce | W<br>Sonce | Write<br>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. |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.12.1 CPUSYSLOCK1 Register (Offset = 0h) [Reset = 00000000h]

CPUSYSLOCK1 is shown in Figure 3-161 and described in Table 3-178.

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 3-161. CPUSYSLOCK1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIOLPMSEL1 | GPIOLPMSEL0 | LPMCR | SECMSEL | PCLKCR16 | PCLKCR15 | PCLKCR14 | PCLKCR13 |
| R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| PCLKCR12 | PCLKCR11 | PCLKCR10 | PCLKCR9 | PCLKCR8 | PCLKCR7 | PCLKCR6 | PCLKCR5 |
| R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PCLKCR4 | PCLKCR3 | PCLKCR2 | PCLKCR1 | PCLKCR0 | PIEVERRADDR | IORESTOREADDR | HIBBOOTMODE |
| R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h |

**Table 3-178. CPUSYSLOCK1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-24 | RESERVED | R-0 | 0h | Reserved |
| 23 | GPIOLPMSEL1 | R/WSonce | 0h | Lock bit for GPIOLPMSEL1 Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: SYSRSn |
| 22 | GPIOLPMSEL0 | R/WSonce | 0h | Lock bit for GPIOLPMSEL0 Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: SYSRSn |
| 21 | LPMCR | R/WSonce | 0h | Lock bit for LPMCR Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: SYSRSn |
| 20 | SECMSEL | R/WSonce | 0h | Lock bit for SECMSEL Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: SYSRSn |
| 19 | PCLKCR16 | R/WSonce | 0h | Lock bit for PCLKCR16 Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: SYSRSn |
| 18 | PCLKCR15 | R/WSonce | 0h | Lock bit for PCLKCR15 Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-178. CPUSYSLOCK1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 17 | PCLKCR14 | R/WSonce | 0h | Lock bit for PCLKCR14 Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: SYSRSn |
| 16 | PCLKCR13 | R/WSonce | 0h | Lock bit for PCLKCR13 Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: SYSRSn |
| 15 | PCLKCR12 | R/WSonce | 0h | Lock bit for PCLKCR12 Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: SYSRSn |
| 14 | PCLKCR11 | R/WSonce | 0h | Lock bit for PCLKCR11 Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: SYSRSn |
| 13 | PCLKCR10 | R/WSonce | 0h | Lock bit for PCLKCR10 Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: SYSRSn |
| 12 | PCLKCR9 | R/WSonce | 0h | Lock bit for PCLKCR9 Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: SYSRSn |
| 11 | PCLKCR8 | R/WSonce | 0h | Lock bit for PCLKCR8 Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: SYSRSn |
| 10 | PCLKCR7 | R/WSonce | 0h | Lock bit for PCLKCR7 Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: SYSRSn |
| 9 | PCLKCR6 | R/WSonce | 0h | Lock bit for PCLKCR6 Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: SYSRSn |
| 8 | PCLKCR5 | R/WSonce | 0h | Lock bit for PCLKCR5 Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: SYSRSn |
| 7 | PCLKCR4 | R/WSonce | 0h | Lock bit for PCLKCR4 Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: SYSRSn |
| 6 | PCLKCR3 | R/WSonce | 0h | Lock bit for PCLKCR3 Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: SYSRSn |
| 5 | PCLKCR2 | R/WSonce | 0h | Lock bit for PCLKCR2 Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: SYSRSn |
| 4 | PCLKCR1 | R/WSonce | 0h | Lock bit for PCLKCR1 Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-178. CPUSYSLOCK1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3 | PCLKCR0 | R/WSonce | 0h | Lock bit for PCLKCR0 Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: SYSRSn |
| 2 | PIEVERRADDR | R/WSonce | 0h | Lock bit for PIEVERRADDR Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: SYSRSn |
| 1 | IORESTOREADDR | R/WSonce | 0h | Lock bit for IORESTOREADDR Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: SYSRSn |
| 0 | HIBBOOTMODE | R/WSonce | 0h | Lock bit for HIBBOOTMODE register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.12.2 HIBBOOTMODE Register (Offset = 6h) [Reset = 0000000Fh]

HIBBOOTMODE is shown in Figure 3-162 and described in Table 3-179.

Return to the Summary Table.

HIB Boot Mode Register

**Figure 3-162. HIBBOOTMODE Register**

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

**Table 3-179. HIBBOOTMODE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | BMODE | R/W | Fh | This register defined the boot mode on a HIB Wakeup. Its the responsibility of user to initialize the appropriate boot mode before going into HIB mode. Refer to the Boot ROM section for more details on this register<br>Reset type: POR |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.12.3 IORESTOREADDR Register (Offset = 8h) [Reset = 003FFFFFh]

IORESTOREADDR is shown in Figure 3-163 and described in Table 3-180.

Return to the Summary Table.

IORestore() routine Address Register

**Figure 3-163. IORESTOREADDR Register**

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

**Table 3-180. IORESTOREADDR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-22 | RESERVED | R-0 | 0h | Reserved |
| 21-0 | ADDR | R/W | 003FFFFFh | This register defines the address of the restoreIO() routine on a HIB wakeup. Its the responsibility of user to initialize this register with the restoreIO() routine address before going into HIB mode. Refer to the Boot ROM section for more details on this register.<br>Reset type: POR |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.12.4 PIEVERRADDR Register (Offset = Ah) [Reset = 003FFFFFh]

PIEVERRADDR is shown in Figure 3-164 and described in Table 3-181.

Return to the Summary Table.

PIE Vector Fetch Error Address register

**Figure 3-164. 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-003FFFFFh | | | | | | | | | | | | | | | | | | | | | |

**Table 3-181. PIEVERRADDR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-22 | RESERVED | R-0 | 0h | Reserved |
| 21-0 | ADDR | R/W | 003FFFFFh | 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 0x3fffbe will get executed. Refer to the Boot ROM section for more details on this register.<br>Reset type: XRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.12.5 PCLKCR0 Register (Offset = 22h) [Reset = 00000038h]

PCLKCR0 is shown in Figure 3-165 and described in Table 3-182.

Return to the Summary Table.

Peripheral Clock Gating Registers

**Figure 3-165. PCLKCR0 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | GTBCLKSYNC | TBCLKSYNC | RESERVED | HRPWM |
| R-0-0h | | | | R/W-0h | R/W-0h | R-0-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | CPUTIMER2 | CPUTIMER1 | CPUTIMER0 | DMA | RESERVED | CLA1 |
| R-0-0h | | R/W-1h | R/W-1h | R/W-1h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-182. PCLKCR0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-20 | RESERVED | R-0 | 0h | Reserved |
| 19 | GTBCLKSYNC | R/W | 0h | EPWM Time Base Clock Global sync: When set by CPU1, PWM time bases of all modules start counting. The effect of this bit is seen on all the EPMW modules irrespective of their partitioning based on CPUSEL<br>Notes:<br>1. This bit on the CPU2.PCLKCR0 register has no effect.<br>2. Writing '1' to this bit overrides the effect of write '1' to the TBCLKSYNC bit at the same time<br>Reset type: SYSRSn |
| 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. This bit only impacts the TBCTR of all EPWM modules. Everything except the TBCTR of each module enabled in PCLKCR2 will still be clocked by EPWMCLK.<br>Notes:<br>1. This bit from CPU1.PCLKCR0 or CPU2.PCLKCR0 is selected and fed to the individual EPWM modules based on their respective CPUSEL bit.<br>Reset type: SYSRSn |
| 17 | RESERVED | R-0 | 0h | Reserved |
| 16 | HRPWM | R/W | 0h | HRPWM Clock Enable Bit: When set, this enables the clock to the HRPWM module<br>1: HRPWM clock is enabled<br>0: HRPWM clock is disabled<br>Note:<br>[1] This bit is present only in CPU1.PCLKCR0. This bit is not used (R/W) in CPU2.PCLKCR0<br>Reset type: SYSRSn |
| 15-6 | RESERVED | R-0 | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-182. PCLKCR0 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 5 | CPUTIMER2 | R/W | 1h | CPUTIMER2 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 4 | CPUTIMER1 | R/W | 1h | CPUTIMER1 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 3 | CPUTIMER0 | R/W | 1h | CPUTIMER0 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 2 | DMA | R/W | 0h | DMA Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 1 | RESERVED | R/W | 0h | Reserved |
| 0 | CLA1 | R/W | 0h | CLA1 Clock Enable Bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.12.6 PCLKCR1 Register (Offset = 24h) [Reset = 00000000h]

PCLKCR1 is shown in Figure 3-166 and described in Table 3-183.

Return to the Summary Table.

Peripheral Clock Gating Registers

**Figure 3-166. PCLKCR1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | EMIF2 | EMIF1 |
| R-0-0h | | | | | | R/W-0h | R/W-0h |

**Table 3-183. PCLKCR1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-2 | RESERVED | R-0 | 0h | Reserved |
| 1 | EMIF2 | R/W | 0h | EMIF2 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Notes:<br>[1] These bits are not used (R/W) in CPU2.PCLKCR1 register. EMIF1 & EMIF2 clock enabled are controlled only from CPU1.PCLKCR1 register.<br>Reset type: SYSRSn |
| 0 | EMIF1 | R/W | 0h | EMIF1 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Notes:<br>[1] These bits are not used (R/W) in CPU2.PCLKCR1 register. EMIF1 & EMIF2 clock enabled are controlled only from CPU1.PCLKCR1 register.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.12.7 PCLKCR2 Register (Offset = 26h) [Reset = 00000000h]

PCLKCR2 is shown in Figure 3-167 and described in Table 3-184.

Return to the Summary Table.

Peripheral Clock Gating Registers

**Figure 3-167. PCLKCR2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | RESERVED | RESERVED | RESERVED | EPWM12 | EPWM11 | EPWM10 | EPWM9 |
| R/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 |
| EPWM8 | EPWM7 | EPWM6 | EPWM5 | EPWM4 | EPWM3 | EPWM2 | EPWM1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-184. 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 | EPWM12 | R/W | 0h | EPWM12 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 10 | EPWM11 | R/W | 0h | EPWM11 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 9 | EPWM10 | R/W | 0h | EPWM10 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 8 | EPWM9 | R/W | 0h | EPWM9 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 7 | EPWM8 | R/W | 0h | EPWM8 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 6 | EPWM7 | R/W | 0h | EPWM7 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-184. PCLKCR2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 5 | EPWM6 | R/W | 0h | EPWM6 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 4 | EPWM5 | R/W | 0h | EPWM5 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 3 | EPWM4 | R/W | 0h | EPWM4 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Notes:<br>[1] This bit is also used to enable clocking for CLB4<br>Reset type: SYSRSn |
| 2 | EPWM3 | R/W | 0h | EPWM3 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Notes:<br>[1] This bit is also used to enable clocking for CLB3<br>Reset type: SYSRSn |
| 1 | EPWM2 | R/W | 0h | EPWM2 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Notes:<br>[1] This bit is also used to enable clocking for CLB2<br>Reset type: SYSRSn |
| 0 | EPWM1 | R/W | 0h | EPWM1 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Notes:<br>[1] This bit is also used to enable clocking for CLB1<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.12.8 PCLKCR3 Register (Offset = 28h) [Reset = 00000000h]

PCLKCR3 is shown in Figure 3-168 and described in Table 3-185.

Return to the Summary Table.

Peripheral Clock Gating Registers

**Figure 3-168. PCLKCR3 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | RESERVED | ECAP6 | ECAP5 | ECAP4 | ECAP3 | ECAP2 | ECAP1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-185. PCLKCR3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-8 | RESERVED | R-0 | 0h | Reserved |
| 7 | RESERVED | R/W | 0h | Reserved |
| 6 | RESERVED | R/W | 0h | Reserved |
| 5 | ECAP6 | R/W | 0h | ECAP6 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 4 | ECAP5 | R/W | 0h | ECAP5 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 3 | ECAP4 | R/W | 0h | ECAP4 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 2 | ECAP3 | R/W | 0h | ECAP3 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 1 | ECAP2 | R/W | 0h | ECAP2 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 0 | ECAP1 | R/W | 0h | ECAP1 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.12.9 PCLKCR4 Register (Offset = 2Ah) [Reset = 00000000h]

PCLKCR4 is shown in Figure 3-169 and described in Table 3-186.

Return to the Summary Table.

Peripheral Clock Gating Registers

**Figure 3-169. PCLKCR4 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | RESERVED | EQEP3 | EQEP2 | EQEP1 |
| R-0-0h | | | | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-186. PCLKCR4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-4 | RESERVED | R-0 | 0h | Reserved |
| 3 | RESERVED | R/W | 0h | Reserved |
| 2 | EQEP3 | R/W | 0h | EQEP3 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 1 | EQEP2 | R/W | 0h | EQEP2 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 0 | EQEP1 | R/W | 0h | EQEP1 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.12.10 PCLKCR6 Register (Offset = 2Eh) [Reset = 00000000h]

PCLKCR6 is shown in Figure 3-170 and described in Table 3-187.

Return to the Summary Table.

Peripheral Clock Gating Registers

**Figure 3-170. PCLKCR6 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | |
| R-0-0h | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | | | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | SD2 | SD1 |
| R-0-0h | | | | | | | | 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. PCLKCR6 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-8 | RESERVED | R-0 | 0h | Reserved |
| 7 | RESERVED | R/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 | SD2 | R/W | 0h | SD2 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 0 | SD1 | R/W | 0h | SD1 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.12.11 PCLKCR7 Register (Offset = 30h) [Reset = 00000000h]

PCLKCR7 is shown in Figure 3-171 and described in Table 3-188.

Return to the Summary Table.

Peripheral Clock Gating Registers

**Figure 3-171. PCLKCR7 Register**

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

**Table 3-188. PCLKCR7 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-4 | RESERVED | R-0 | 0h | Reserved |
| 3 | SCI_D | R/W | 0h | SCI_D Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 2 | SCI_C | R/W | 0h | SCI_C Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 1 | SCI_B | R/W | 0h | SCI_B Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 0 | SCI_A | R/W | 0h | SCI_A Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.12.12 PCLKCR8 Register (Offset = 32h) [Reset = 00000000h]

PCLKCR8 is shown in Figure 3-172 and described in Table 3-189.

Return to the Summary Table.

Peripheral Clock Gating Registers

**Figure 3-172. 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 | SPI_C | SPI_B | SPI_A |
| R-0-0h | | | | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-189. 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 | SPI_C | R/W | 0h | SPI_C Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 1 | SPI_B | R/W | 0h | SPI_B Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 0 | SPI_A | R/W | 0h | SPI_A Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.12.13 PCLKCR9 Register (Offset = 34h) [Reset = 00000000h]

PCLKCR9 is shown in Figure 3-173 and described in Table 3-190.

Return to the Summary Table.

Peripheral Clock Gating Registers

**Figure 3-173. PCLKCR9 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | RESERVED | RESERVED |
| R-0-0h | | | | | | R/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 | | | | | | I2C_B | I2C_A |
| R-0-0h | | | | | | R/W-0h | R/W-0h |

**Table 3-190. PCLKCR9 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 |
| 15-2 | RESERVED | R-0 | 0h | Reserved |
| 1 | I2C_B | R/W | 0h | I2C_B Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 0 | I2C_A | R/W | 0h | I2C_A Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.12.14 PCLKCR10 Register (Offset = 36h) [Reset = 00000000h]

PCLKCR10 is shown in Figure 3-174 and described in Table 3-191.

Return to the Summary Table.

Peripheral Clock Gating Registers

**Figure 3-174. PCLKCR10 Register**

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

**Table 3-191. PCLKCR10 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-4 | RESERVED | R-0 | 0h | Reserved |
| 3 | RESERVED | R/W | 0h | Reserved |
| 2 | RESERVED | R/W | 0h | Reserved |
| 1 | CAN_B | R/W | 0h | CAN_B Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 0 | CAN_A | R/W | 0h | CAN_A Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.12.15 PCLKCR11 Register (Offset = 38h) [Reset = 00000000h]

PCLKCR11 is shown in Figure 3-175 and described in Table 3-192.

Return to the Summary Table.

Peripheral Clock Gating Registers

**Figure 3-175. PCLKCR11 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | RESERVED | USB_A |
| R-0-0h | | | | | | R/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 | | | | | | McBSP_B | McBSP_A |
| R-0-0h | | | | | | R/W-0h | R/W-0h |

**Table 3-192. PCLKCR11 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-18 | RESERVED | R-0 | 0h | Reserved |
| 17 | RESERVED | R/W | 0h | Reserved |
| 16 | USB_A | R/W | 0h | USB_A Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Notes:<br>[1] This bit is not used (R/W) in CPU2.PCLKCR11 register. USB_A clock enabled is controlled only from CPU1.PCLKCR11 register<br>Reset type: SYSRSn |
| 15-2 | RESERVED | R-0 | 0h | Reserved |
| 1 | McBSP_B | R/W | 0h | McBSP_B Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 0 | McBSP_A | R/W | 0h | McBSP_A Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.12.16 PCLKCR12 Register (Offset = 3Ah) [Reset = 00000000h]

PCLKCR12 is shown in Figure 3-176 and described in Table 3-193.

Return to the Summary Table.

Peripheral Clock Gating Registers

**Figure 3-176. PCLKCR12 Register**

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

**Table 3-193. PCLKCR12 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-2 | RESERVED | R-0 | 0h | Reserved |
| 1 | RESERVED | R/W | 0h | Reserved |
| 0 | uPP_A | R/W | 0h | uPP_A Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Notes:<br>[1]] This bit also affects the uPP message RAM wrapper associated with the respective uPP module<br>[2] This bit is not used (R/W) in CPU2.PCLKCR12 register. UPP_A clock enabled is controlled only from CPU1.PCLKCR12 register<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.12.17 PCLKCR13 Register (Offset = 3Ch) [Reset = 00000000h]

PCLKCR13 is shown in Figure 3-177 and described in Table 3-194.

Return to the Summary Table.

Peripheral Clock Gating Registers

**Figure 3-177. PCLKCR13 Register**

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

**Table 3-194. PCLKCR13 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-4 | RESERVED | R-0 | 0h | Reserved |
| 3 | ADC_D | R/W | 0h | ADC_D Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 2 | ADC_C | R/W | 0h | ADC_C Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 1 | ADC_B | R/W | 0h | ADC_B Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 0 | ADC_A | R/W | 0h | ADC_A Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.12.18 PCLKCR14 Register (Offset = 3Eh) [Reset = 00000000h]

PCLKCR14 is shown in Figure 3-178 and described in Table 3-195.

Return to the Summary Table.

Peripheral Clock Gating Registers

**Figure 3-178. PCLKCR14 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CMPSS8 | CMPSS7 | CMPSS6 | CMPSS5 | CMPSS4 | CMPSS3 | CMPSS2 | CMPSS1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-195. PCLKCR14 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-8 | RESERVED | R-0 | 0h | Reserved |
| 7 | CMPSS8 | R/W | 0h | CMPSS8 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 6 | CMPSS7 | R/W | 0h | CMPSS7 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 5 | CMPSS6 | R/W | 0h | CMPSS6 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 4 | CMPSS5 | R/W | 0h | CMPSS5 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 3 | CMPSS4 | R/W | 0h | CMPSS4 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 2 | CMPSS3 | R/W | 0h | CMPSS3 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 1 | CMPSS2 | R/W | 0h | CMPSS2 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-195. PCLKCR14 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | CMPSS1 | R/W | 0h | CMPSS1 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.12.19 PCLKCR16 Register (Offset = 42h) [Reset = 00000000h]

PCLKCR16 is shown in Figure 3-179 and described in Table 3-196.

Return to the Summary Table.

Peripheral Clock Gating Registers

**Figure 3-179. PCLKCR16 Register**

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

**Table 3-196. PCLKCR16 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-20 | RESERVED | R-0 | 0h | Reserved |
| 19 | RESERVED | R/W | 0h | Reserved |
| 18 | DAC_C | R/W | 0h | Buffered_DAC_C Clock Enable Bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 17 | DAC_B | R/W | 0h | Buffered_DAC_B Clock Enable Bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 16 | DAC_A | R/W | 0h | Buffered_DAC_A Clock Enable Bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 15-4 | RESERVED | R-0 | 0h | Reserved |
| 3 | RESERVED | R/W | 0h | Reserved |
| 2 | RESERVED | R/W | 0h | Reserved |
| 1 | RESERVED | R/W | 0h | Reserved |
| 0 | RESERVED | R/W | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.12.20 SECMSEL Register (Offset = 74h) [Reset = 00000000h]

SECMSEL is shown in Figure 3-180 and described in Table 3-197.

Return to the Summary Table.

Secondary Master Select register for common peripherals: Selects between CLA & DMA

**Figure 3-180. SECMSEL Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | RESERVED | | RESERVED | | RESERVED | |
| R-0-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | RESERVED | | PF2SEL | | PF1SEL | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 3-197. SECMSEL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-14 | RESERVED | R-0 | 0h | Reserved |
| 13-12 | RESERVED | R/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 | PF2SEL | R/W | 0h | This bit selects whether the dual ported bridge is connected with DMA or CLA as the secondary master (C28x is always connected as primary master)<br>x0: Bridge is connected to CLA<br>x1: Bridge is connected to DMA<br>Reset type: SYSRSn |
| 1-0 | PF1SEL | R/W | 0h | This bit selects whether the dual ported bridge is connected with DMA or CLA as the secondary master (C28x is always connected as primary master)<br>x0: Bridge is connected to CLA<br>x1: Bridge is connected to DMA<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.12.21 LPMCR Register (Offset = 76h) [Reset = 000000FCh]

LPMCR is shown in Figure 3-181 and described in Table 3-198.

Return to the Summary Table.

LPM Control Register

**Figure 3-181. LPMCR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| IOISODIS | RESERVED | | | | | | |
| R/W1S-0h | R-0-0h | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | M0M1MODE | |
| R-0-0h | | | | | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| WDINTE | RESERVED | | | | | | |
| R/W-0h | R-0-0h | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| QUALSTDBY | | | | | | LPM | |
| R/W-3Fh | | | | | | R/W-0h | |

**Table 3-198. LPMCR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | IOISODIS | R/W1S | 0h | 0: Indicates IO ISOLATION is not turned ON<br>1: Indicates IO ISOLATION is turned ON. This bit is set one by hardware ONLY during HIB. This bit cant be set to 1 by software<br>Writing 0 to this bit has not effect.<br>Writing 1 to this bit deactivates IO ISOLATION<br>Notes:<br>[1] This bit is reserved in the register mapped to CPU2<br>Reset type: raw-XRSn |
| 30-18 | RESERVED | R-0 | 0h | Reserved |
| 17-16 | M0M1MODE | R/W | 0h | These bit control the state of CPU1's and CPU2's M0 & M1 memories when Device goes into HIB mode.<br>00: CPUx's M0 & M1 memories ON with low-leakage mode<br>01: CPUx's M0 & M1 memories OFF<br>1x: Reserved<br>Notes:<br>[1] Low-leakage mode for M0 & M1 memories uses the 'Retention' feature of the SRAMs.<br>[2] These bits take effect only when device goes into HIB mode. If the device is not in HIB mode, the value in this bit doesn't control the state of CPU1's and CPU2's M0 & M1 memories<br>Reset type: POR |
| 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.<br>Note:<br>[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<br>Reset type: SYSRSn |
| 14-8 | RESERVED | R-0 | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-198. LPMCR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 7-2 | QUALSTDBY | R/W | 3Fh | Select number of OSCCLK clock cycles to qualify the selected inputs when waking the from STANDBY mode:<br>000000 = 2 OSCCLKs<br>000001 = 3 OSCCLKs<br>.....<br>111111 = 65 OSCCLKs<br>Note: The LPMCR.QUALSTDBY register should be set to a value greater than the ratio of INTOSC1/PLLSYSCLK to ensure proper wake up.<br>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)<br>00: IDLE Mode<br>01: STANDBY Mode<br>10: HALT Mode (treated as STANDBY for CPU2)<br>11: HIB Mode (treated as STANDBY for CPU2)<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.12.22 GPIOLPMSEL0 Register (Offset = 78h) [Reset = 00000000h]

GPIOLPMSEL0 is shown in Figure 3-182 and described in Table 3-199.

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 3-182. GPIOLPMSEL0 Register**

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

**Table 3-199. GPIOLPMSEL0 Register Field Descriptions**

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



Copyright © 2024 Texas Instruments Incorporated

**Table 3-199. GPIOLPMSEL0 Register Field Descriptions (continued)**

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



Copyright © 2024 Texas Instruments Incorporated

**Table 3-199. GPIOLPMSEL0 Register Field Descriptions (continued)**

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



Copyright © 2024 Texas Instruments Incorporated

### 3.17.12.23 GPIOLPMSEL1 Register (Offset = 7Ah) [Reset = 00000000h]

GPIOLPMSEL1 is shown in Figure 3-183 and described in Table 3-200.

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 3-183. GPIOLPMSEL1 Register**

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

**Table 3-200. GPIOLPMSEL1 Register Field Descriptions**

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



Copyright © 2024 Texas Instruments Incorporated

**Table 3-200. GPIOLPMSEL1 Register Field Descriptions (continued)**

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



Copyright © 2024 Texas Instruments Incorporated

**Table 3-200. GPIOLPMSEL1 Register Field Descriptions (continued)**

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



Copyright © 2024 Texas Instruments Incorporated

### 3.17.12.24 TMR2CLKCTL Register (Offset = 7Ch) [Reset = 00000000h]

TMR2CLKCTL is shown in Figure 3-184 and described in Table 3-201.

Return to the Summary Table.

Timer2 Clock Measurement functionality control register
This memory mapped register requires a delay of 69 SYSCLK cycles between subsequent writes to the register, otherwise a second write can be lost. This delay can be realized by adding 69 NOP instructions.

**Figure 3-184. 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 3-201. TMR2CLKCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-6 | RESERVED | R-0 | 0h | Reserved |
| 5-3 | TMR2CLKPRESCALE | R/W | 0h | CPU Timer 2 Clock Pre-Scale Value: These bits select the pre-scale value for the selected clock source for CPU Timer 2:<br>0,0,0,/1 (default on reset)<br>0,0,1,/2,<br>0,1,0,/4<br>0,1,1,/8<br>1,0,0,/16<br>1,0,1,spare (defaults to /16)<br>1,1,0,spare (defaults to /16)<br>1,1,1,spare (defaults to /16)<br>Note:<br>[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.<br>[2] Pre-scaler is bypassed if SYSCLK is selected as the source of CPU Timer 2 in TMR2CLKSRCSEL of TMR2CLKCTL.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-201. TMR2CLKCTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 2-0 | TMR2CLKSRCSEL | R/W | 0h | CPU Timer 2 Clock Source Select Bit: This bit selects the source for CPU Timer 2:<br>000 =SYSCLK Selected (default on reset, pre-scale is bypassed)<br>001 = INTOSC1<br>010 = INTOSC2<br>011 = XTAL<br>100 = Reserved<br>101 = Reserved<br>110 = AUXPLLCLK<br>111 = reserved<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.12.25 RESC Register (Offset = 80h) [Reset = 00000003h]

RESC is shown in Figure 3-185 and described in Table 3-202.

Return to the Summary Table.

Reset Cause register

**Figure 3-185. RESC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| TRSTn_pin_status | XRSn_pin_status | RESERVED | | | | | |
| R-0h | 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 |
| RESERVED | | | | | | | SCCRESETn |
| R-0-0h | | | | | | | R/W1C-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | HIBRESETn | HWBISTn | RESERVED | NMIWDRSn | WDRSn | XRSn | POR |
| R-0-0h | R/W1C-0h | R/W1C-0h | R-0-0h | R/W1C-0h | R/W1C-0h | R/W1C-1h | R/W1C-1h |

**Table 3-202. RESC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | TRSTn_pin_status | R | 0h | Reading this bit provides the current status of TRSTn at the respective C28x CPUs TRSTn input port. Reset value is reflective of the pin status.<br>Reset type: N/A |
| 30 | XRSn_pin_status | R | 0h | Reading this bit provides the current status of the XRSn pin. Reset value is reflective of the pin status.<br>Reset type: N/A |
| 29-16 | RESERVED | R-0 | 0h | Reserved |
| 15-9 | RESERVED | R-0 | 0h | Reserved |
| 8 | SCCRESETn | R/W1C | 0h | If this bit is set, indicates that the device was reset by SCCRESETn (fired by DCSM).<br>Writing a 1 to this bit will force the bit to 0<br>Writing of 0 will have no effect.<br>Reset type: POR |
| 7 | RESERVED | R-0 | 0h | Reserved |
| 6 | HIBRESETn | R/W1C | 0h | If this bit is set, indicates that the device was reset due to a Hibernate mode Wakeup.<br>Writing a 1 to this bit will force the bit to 0<br>Writing of 0 will have no effect.<br>Reset type: raw-XRSn |
| 5 | HWBISTn | R/W1C | 0h | If this bit is set, indicates that the device was reset by HWBIST.<br>Writing a 1 to this bit will force the bit to 0<br>Writing of 0 will have no effect.<br>Reset type: POR |
| 4 | RESERVED | R-0 | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-202. RESC Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3 | NMIWDRSn | R/W1C | 0h | If this bit is set, indicates that the device was reset by NMIWDRSn.<br>Writing a 1 to this bit will force the bit to 0<br>Writing of 0 will have no effect.<br>To know the exact cause of NMI after the reset, software needs to read CPU1/2.NMISHDFLG registers<br>Reset type: POR |
| 2 | WDRSn | R/W1C | 0h | If this bit is set, indicates that the device was reset by WDRSn.<br>Writing a 1 to this bit will force the bit to 0<br>Writing of 0 will have no effect.<br>Reset type: POR |
| 1 | XRSn | R/W1C | 1h | If this bit is set, indicates that the device was reset by XRSn.<br>Writing a 1 to this bit will force the bit to 0<br>Writing of 0 will have no effect.<br>Reset type: POR |
| 0 | POR | R/W1C | 1h | If this bit is set, indicates that the device was reset by PORn. This bit is cleared by the Boot-ROM. As such, this bit cannot be used by the application to determine a POR.<br>Writing a 1 to this bit will force the bit to 0<br>Writing of 0 will have no effect.<br>Reset type: POR |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.13 ROM_PREFETCH_REGS Registers

Table 3-203 lists the memory-mapped registers for the ROM_PREFETCH_REGS registers. All register offset addresses not listed in Table 3-203 should be considered as reserved locations and the register contents should not be modified.

**Table 3-203. ROM_PREFETCH_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | ROMPREFETCH | ROM Prefetch Configuration Register | EALLOW | Go |

Complex bit access types are encoded to fit into small table cells. Table 3-204 shows the codes that are used for access types in this section.

**Table 3-204. ROM_PREFETCH_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. |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.13.1 ROMPREFETCH Register (Offset = 0h) [Reset = 00000000h]

ROMPREFETCH is shown in Figure 3-186 and described in Table 3-205.

Return to the Summary Table.

ROM Prefetch Configuration Register

**Figure 3-186. ROMPREFETCH Register**

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

**Table 3-205. ROMPREFETCH Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-1 | RESERVED | R | 0h | Reserved |
| 0 | PFENABLE | R/W | 0h | 0: Prefetch is disabled for secure ROM and boot ROM.<br>1: Prefetch is enabled for secure ROM and boot ROM.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.14 DCSM_Z1_REGS Registers

Table 3-206 lists the memory-mapped registers for the DCSM_Z1_REGS registers. All register offset addresses not listed in Table 3-206 should be considered as reserved locations and the register contents should not be modified.

**Table 3-206. DCSM_Z1_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | Z1_LINKPOINTER | Zone 1 Link Pointer | | Go |
| 2h | Z1_OTPSECLOCK | Zone 1 OTP Secure JTAG lock | | Go |
| 4h | Z1_BOOTCTRL | Boot Mode | | Go |
| 6h | Z1_LINKPOINTERERR | Link Pointer Error | | Go |
| 10h | Z1_CSMKEY0 | Zone 1 CSM Key 0 | | Go |
| 12h | Z1_CSMKEY1 | Zone 1 CSM Key 1 | | Go |
| 14h | Z1_CSMKEY2 | Zone 1 CSM Key 2 | | Go |
| 16h | Z1_CSMKEY3 | Zone 1 CSM Key 3 | | Go |
| 19h | Z1_CR | Zone 1 CSM Control Register | | Go |
| 1Ah | Z1_GRABSECTR | Zone 1 Grab Flash Sectors Register | | Go |
| 1Ch | Z1_GRABRAMR | Zone 1 Grab RAM Blocks Register | | Go |
| 1Eh | Z1_EXEONLYSECTR | Zone 1 Flash Execute_Only Sector Register | | Go |
| 20h | Z1_EXEONLYRAMR | Zone 1 RAM Execute_Only Block Register | | Go |

Complex bit access types are encoded to fit into small table cells. Table 3-207 shows the codes that are used for access types in this section.

**Table 3-207. DCSM_Z1_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| R-0 | R<br>-0 | Read<br>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. |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.14.1 Z1_LINKPOINTER Register (Offset = 0h) [Reset = E0000000h]

Z1_LINKPOINTER is shown in Figure 3-187 and described in Table 3-208.

Return to the Summary Table.

Zone 1 Link Pointer

**Figure 3-187. Z1_LINKPOINTER Register**

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

**Table 3-208. Z1_LINKPOINTER Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-29 | RESERVED | R | 7h | Reserved |
| 28-0 | LINKPOINTER | R | 0h | This is resolved Link-Pointer value which is generated by looking at the three physical Link-Pointer values loaded from OTP.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.14.2 Z1_OTPSECLOCK Register (Offset = 2h) [Reset = 00000FFFh]

Z1_OTPSECLOCK is shown in Figure 3-188 and described in Table 3-209.

Return to the Summary Table.

Zone 1 OTP Secure JTAG lock

**Figure 3-188. Z1_OTPSECLOCK Register**

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

**Table 3-209. Z1_OTPSECLOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-12 | RESERVED | R | 0h | Reserved |
| 11-8 | CRCLOCK | R | Fh | Value in this field gets loaded from Z1_CRCLOCK[3:0] when a read is issued to address location of Z1_CRCLOCK in OTP.<br>1111 : VCU has ability to calculate CRC on secure memories.<br>Other Value : VCU doesn't have ability to calculate CRC on secure memories.<br>Reset type: SYSRSn |
| 7-4 | PSWDLOCK | R | Fh | Value in this field gets loaded from Z1_PSWDLOCK[3:0] when a read is issued to address location of Z1_PSWDLOCK in OTP.<br>1111 : CSM password locations in OTP are not protected and can be read from debugger as well as code running from anywhere.<br>Other Value : CSM password locations in OTP are protected and can't be read without unlocking CSM of that zone.<br>Reset type: SYSRSn |
| 3-0 | JTAGLOCK | R | Fh | Value in this field gets loaded from Z1_JTAGLOCK[3:0] when a read is issued to address location of Z1_JTAGLOCK in OTP.<br>1111 : JTAG/Emulation access is allowed.<br>Other Value : JTAG/Emulation access not allowed.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.14.3 Z1_BOOTCTRL Register (Offset = 4h) [Reset = 00000000h]

Z1_BOOTCTRL is shown in Figure 3-189 and described in Table 3-210.

Return to the Summary Table.

Boot Mode

**Figure 3-189. Z1_BOOTCTRL Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BOOTPIN1 | | | | | | | | BOOTPIN0 | | | | | | | | BMODE | | | | | | | | KEY | | | | | | | |
| R-0h | | | | | | | | R-0h | | | | | | | | R-0h | | | | | | | | R-0h | | | | | | | |

**Table 3-210. Z1_BOOTCTRL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-24 | BOOTPIN1 | R | 0h | This field gets loaded with Z1_BOOTCTRL[31:24] when a dummy read is issued to address location of Z1_BOOTCTRL in OTP.<br>This assigns the pin to be used as BOOTPIN1.<br>0 : Pick default bootmode pin.<br>1 : Pick GPIO0 as BOOTPIN1.<br>2 : Pick GPIO1 as BOOTPIN1.<br>....<br>....<br>n : Pick GPIOn-1 as BOOTPIN1.<br>Reset type: SYSRSn |
| 23-16 | BOOTPIN0 | R | 0h | This field gets loaded with Z1_BOOTCTRL[23:16] when a dummy read is issued to address location of Z1_BOOTCTRL in OTP.<br>This assigns the pin to be used as BOOTPIN1.<br>0 : Pick default bootmode pin.<br>1 : Pick GPIO0 as BOOTPIN1.<br>2 : Pick GPIO1 as BOOTPIN1.<br>....<br>....<br>n : Pick GPIOn-1 as BOOTPIN1.<br>Reset type: SYSRSn |
| 15-8 | BMODE | R | 0h | This field gets loaded with Z1_BOOTCTRL[15:8] when a dummy read is issued to address location of Z1_BOOTCTRL in OTP.<br>Reset type: SYSRSn |
| 7-0 | KEY | R | 0h | This field gets loaded with Z1_BOOTCTRL[7:0] when a dummy read is issued to address location of Z1_BOOTCTRL in OTP.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.14.4 Z1_LINKPOINTERERR Register (Offset = 6h) [Reset = FFFFFFFFh]

Z1_LINKPOINTERERR is shown in Figure 3-190 and described in Table 3-211.

Return to the Summary Table.

Link Pointer Error

**Figure 3-190. Z1_LINKPOINTERERR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Z1_LINKPOINTERERR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R-FFFFFFFFh | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 3-211. Z1_LINKPOINTERERR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | Z1_LINKPOINTERERR | R | FFFFFFFFh | These bits indicate errors during formation of the resolved Link-Pointer value after the three physical Link-Pointer values loaded of OTP.<br>0 : No Error.<br>Other : Error on bit positions which is set to 1.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.14.5 Z1_CSMKEY0 Register (Offset = 10h) [Reset = 00000000h]

Z1_CSMKEY0 is shown in Figure 3-191 and described in Table 3-212.

Return to the Summary Table.

Zone 1 CSM Key 0

**Figure 3-191. Z1_CSMKEY0 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Z1_CSMKEY0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 3-212. Z1_CSMKEY0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | Z1_CSMKEY0 | R | 0h | To unclock Zone1, user needs to write this regsiter 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.)<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.14.6 Z1_CSMKEY1 Register (Offset = 12h) [Reset = 00000000h]

Z1_CSMKEY1 is shown in Figure 3-192 and described in Table 3-213.

Return to the Summary Table.

Zone 1 CSM Key 1

**Figure 3-192. Z1_CSMKEY1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Z1_CSMKEY1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 3-213. Z1_CSMKEY1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | Z1_CSMKEY1 | R | 0h | To unclock Zone1, user needs to write this regsiter 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.)<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.14.7 Z1_CSMKEY2 Register (Offset = 14h) [Reset = 00000000h]

Z1_CSMKEY2 is shown in Figure 3-193 and described in Table 3-214.

Return to the Summary Table.

Zone 1 CSM Key 2

**Figure 3-193. Z1_CSMKEY2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Z1_CSMKEY2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 3-214. Z1_CSMKEY2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | Z1_CSMKEY2 | R | 0h | To unclock Zone1, user needs to write this regsiter 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.)<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.14.8 Z1_CSMKEY3 Register (Offset = 16h) [Reset = 00000000h]

Z1_CSMKEY3 is shown in Figure 3-194 and described in Table 3-215.

Return to the Summary Table.

Zone 1 CSM Key 3

**Figure 3-194. Z1_CSMKEY3 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Z1_CSMKEY3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 3-215. Z1_CSMKEY3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | Z1_CSMKEY3 | R | 0h | To unclock Zone1, user needs to write this regsiter 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.)<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.14.9 Z1_CR Register (Offset = 19h) [Reset = 0008h]

Z1_CR is shown in Figure 3-195 and described in Table 3-216.

Return to the Summary Table.

Zone 1 CSM Control Register

**Figure 3-195. Z1_CR Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| FORCESEC | RESERVED | | | | | | |
| R-0/W-0h | R-0h | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | ARMED | UNSECURE | ALLONE | ALLZERO | RESERVED | | |
| R-0h | R-0h | R-0h | R-0h | R-1h | R-0h | | |

**Table 3-216. Z1_CR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | FORCESEC | R-0/W | 0h | A write '1' to this fields resets the state of zone. If zone is unlocked, it'll lock(secure) the zone and also resets all the bits in this register.<br>Reset type: SYSRSn |
| 14-8 | RESERVED | R | 0h | Reserved |
| 7 | RESERVED | R | 0h | Reserved |
| 6 | ARMED | R | 0h | 0 : Dummy read to CSM Password locations in OTP hasn't been performed.<br>1 : Dummy read to CSM Password locations in OTP has been performed.<br>Reset type: SYSRSn |
| 5 | UNSECURE | R | 0h | Indiacates the state of Zone.<br>0 : Zone is in lock(secure) state.<br>1 : Zone is in unlock(unsecure) state.<br>Reset type: SYSRSn |
| 4 | ALLONE | R | 0h | Indicates the state of CSM passowrds.<br>0 : CSM Passwords are not all ones.<br>1 : CSM Passwords are all ones and zone is in unlock state.<br>Reset type: SYSRSn |
| 3 | ALLZERO | R | 1h | Indicates the state of CSM passowrds.<br>0 : CSM Passwords are not all zeros.<br>1 : CSM Passwords are all zero and device is permanently locked.<br>Reset type: SYSRSn |
| 2-0 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.14.10 Z1_GRABSECTR Register (Offset = 1Ah) [Reset = 00000000h]

Z1_GRABSECTR is shown in Figure 3-196 and described in Table 3-217.

Return to the Summary Table.

Zone 1 Grab Flash Sectors Register

**Figure 3-196. Z1_GRABSECTR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | RESERVED | | GRAB_SECTN | | GRAB_SECTM | |
| R-0h | | R-0h | | R-0h | | R-0h | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GRAB_SECTL | | GRAB_SECTK | | GRAB_SECTJ | | GRAB_SECTI | |
| R-0h | | R-0h | | R-0h | | R-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GRAB_SECTH | | GRAB_SECTG | | GRAB_SECTF | | GRAB_SECTE | |
| R-0h | | R-0h | | R-0h | | R-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GRAB_SECTD | | GRAB_SECTC | | GRAB_SECTB | | GRAB_SECTA | |
| R-0h | | R-0h | | R-0h | | R-0h | |

**Table 3-217. Z1_GRABSECTR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | RESERVED | R | 0h | Reserved |
| 29-28 | RESERVED | R | 0h | Reserved |
| 27-26 | GRAB_SECTN | R | 0h | Value in this field gets loaded from Z1_GRABSECT[27:26] when a read is issued to address location of Z1_GRABSECT in OTP.<br>00 : Invalid. Flash Sector N is inaccessible.<br>01 : Request to allocate Flash Sector N to Zone1.<br>10 : Request to allocate Flash Sector N to Zone1.<br>11 : Request to make Flash sector N Non-Secure.<br>Reset type: SYSRSn |
| 25-24 | GRAB_SECTM | R | 0h | Value in this field gets loaded from Z1_GRABSECT[25:24] when a read is issued to address location of Z1_GRABSECT in OTP.<br>00 : Invalid. Flash Sector M is inaccessible.<br>01 : Request to allocate Flash Sector M to Zone1.<br>10 : Request to allocate Flash Sector M to Zone1.<br>11 : Request to make Flash sector M Non-Secure.<br>Reset type: SYSRSn |
| 23-22 | GRAB_SECTL | R | 0h | Value in this field gets loaded from Z1_GRABSECT[23:22] when a read is issued to address location of Z1_GRABSECT in OTP.<br>00 : Invalid. Flash Sector L is inaccessible.<br>01 : Request to allocate Flash Sector L to Zone1.<br>10 : Request to allocate Flash Sector L to Zone1.<br>11 : Request to make Flash sector L Non-Secure.<br>Reset type: SYSRSn |
| 21-20 | GRAB_SECTK | R | 0h | Value in this field gets loaded from Z1_GRABSECT[21:20] when a read is issued to address location of Z1_GRABSECT in OTP.<br>00 : Invalid. Flash Sector K is inaccessible.<br>01 : Request to allocate Flash Sector K to Zone1.<br>10 : Request to allocate Flash Sector K to Zone1.<br>11 : Request to make Flash sector K Non-Secure.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-217. Z1_GRABSECTR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19-18 | GRAB_SECTJ | R | 0h | Value in this field gets loaded from Z1_GRABSECT[19:18] when a read is issued to address location of Z1_GRABSECT in OTP.<br>00 : Invalid. Flash Sector J is inaccessible.<br>01 : Request to allocate Flash Sector J to Zone1.<br>10 : Request to allocate Flash Sector J to Zone1.<br>11 : Request to make Flash sector J Non-Secure.<br>Reset type: SYSRSn |
| 17-16 | GRAB_SECTI | R | 0h | Value in this field gets loaded from Z1_GRABSECT[17:16] when a read is issued to address location of Z1_GRABSECT in OTP.<br>00 : Invalid. Flash Sector I is inaccessible.<br>01 : Request to allocate Flash Sector I to Zone1.<br>10 : Request to allocate Flash Sector I to Zone1.<br>11 : Request to make Flash sector I Non-Secure.<br>Reset type: SYSRSn |
| 15-14 | GRAB_SECTH | R | 0h | Value in this field gets loaded from Z1_GRABSECT[15:14] when a read is issued to address location of Z1_GRABSECT in OTP.<br>00 : Invalid. Flash Sector H is inaccessible.<br>01 : Request to allocate Flash Sector H to Zone1.<br>10 : Request to allocate Flash Sector H to Zone1.<br>11 : Request to make Flash sector H Non-Secure.<br>Reset type: SYSRSn |
| 13-12 | GRAB_SECTG | R | 0h | Value in this field gets loaded from Z1_GRABSECT[13:12] when a read is issued to address location of Z1_GRABSECT in OTP.<br>00 : Invalid. Flash Sector G is inaccessible.<br>01 : Request to allocate Flash Sector G to Zone1.<br>10 : Request to allocate Flash Sector G to Zone1.<br>11 : Request to make Flash sector G Non-Secure.<br>Reset type: SYSRSn |
| 11-10 | GRAB_SECTF | R | 0h | Value in this field gets loaded from Z1_GRABSECT[11:10] when a read is issued to address location of Z1_GRABSECT in OTP.<br>00 : Invalid. Flash Sector F is inaccessible.<br>01 : Request to allocate Flash Sector F to Zone1.<br>10 : Request to allocate Flash Sector F to Zone1.<br>11 : Request to make Flash sector F Non-Secure.<br>Reset type: SYSRSn |
| 9-8 | GRAB_SECTE | R | 0h | Value in this field gets loaded from Z1_GRABSECT[9:8] when a read is issued to address location of Z1_GRABSECT in OTP.<br>00 : Invalid. Flash Sector E is inaccessible.<br>01 : Request to allocate Flash Sector E to Zone1.<br>10 : Request to allocate Flash Sector E to Zone1.<br>11 : Request to make Flash sector E Non-Secure.<br>Reset type: SYSRSn |
| 7-6 | GRAB_SECTD | R | 0h | Value in this field gets loaded from Z1_GRABSECT[7:6] when a read is issued to address location of Z1_GRABSECT in OTP.<br>00 : Invalid. Flash Sector D is inaccessible.<br>01 : Request to allocate Flash Sector D to Zone1.<br>10 : Request to allocate Flash Sector D to Zone1.<br>11 : Request to make Flash sector D Non-Secure.<br>Reset type: SYSRSn |
| 5-4 | GRAB_SECTC | R | 0h | Value in this field gets loaded from Z1_GRABSECT[5:4] when a read is issued to address location of Z1_GRABSECT in OTP.<br>00 : Invalid. Flash Sector C is inaccessible.<br>01 : Request to allocate Flash Sector C to Zone1.<br>10 : Request to allocate Flash Sector C to Zone1.<br>11 : Request to make Flash sector C Non-Secure.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-217. Z1_GRABSECTR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-2 | GRAB_SECTB | R | 0h | Value in this field gets loaded from Z1_GRABSECT[3:2] when a read is issued to address location of Z1_GRABSECT in OTP.<br>00 : Invalid. Flash Sector B is inaccessible.<br>01 : Request to allocate Flash Sector B to Zone1.<br>10 : Request to allocate Flash Sector B to Zone1.<br>11 : Request to make Flash sector B Non-Secure.<br>Reset type: SYSRSn |
| 1-0 | GRAB_SECTA | R | 0h | Value in this field gets loaded from Z1_GRABSECT[1:0] when a read is issued to address location of Z1_GRABSECT in OTP.<br>00 : Invalid. Flash Sector A is inaccessible.<br>01 : Request to allocate Flash Sector A to Zone1.<br>10 : Request to allocate Flash Sector A to Zone1.<br>11 : Request to make Flash sector A Non-Secure.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.14.11 Z1_GRABRAMR Register (Offset = 1Ch) [Reset = 00000000h]

Z1_GRABRAMR is shown in Figure 3-197 and described in Table 3-218.

Return to the Summary Table.

Zone 1 Grab RAM Blocks Register

**Figure 3-197. Z1_GRABRAMR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | GRAB_CLA1 | | RESERVED | | | |
| R-0h | | R-0h | | R-0h | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GRAB_RAM7 | | GRAB_RAM6 | | GRAB_RAM5 | | GRAB_RAM4 | |
| R-0h | | R-0h | | R-0h | | R-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GRAB_RAM3 | | GRAB_RAM2 | | GRAB_RAM1 | | GRAB_RAM0 | |
| R-0h | | R-0h | | R-0h | | R-0h | |

**Table 3-218. Z1_GRABRAMR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | RESERVED | R | 0h | Reserved |
| 29-28 | GRAB_CLA1 | R | 0h | Value in this field gets loaded from Z1_GRABRAM[29:28] when a read is issued to address location of Z1_GRABRAM in OTP.<br>00 : Invalid. CLA1 is inaccessible.<br>01 : Request to allocate CLA1 to Zone1.<br>10 : Request to allocate CLA1 to Zone1.<br>11 : Request to make CLA1 Non-Secure.<br>Reset type: SYSRSn |
| 27-16 | RESERVED | R | 0h | Reserved |
| 15-14 | GRAB_RAM7 | R | 0h | Value in this field gets loaded from Z1_GRABRAM[15:14] when a read is issued to address location of Z1_GRABRAM in OTP.<br>00 : Invalid. D1 RAM is inaccessible.<br>01 : Request to allocate D1 RAM to Zone1.<br>10 : Request to allocate D1 RAM to Zone1.<br>11 : Request to make D1 RAM Non-Secure.<br>Reset type: SYSRSn |
| 13-12 | GRAB_RAM6 | R | 0h | Value in this field gets loaded from Z1_GRABRAM[13:12] when a read is issued to address location of Z1_GRABRAM in OTP.<br>00 : Invalid. D0 RAM is inaccessible.<br>01 : Request to allocate D0 RAM to Zone1.<br>10 : Request to allocate D0 RAM to Zone1.<br>11 : Request to make D0 RAM Non-Secure.<br>Reset type: SYSRSn |
| 11-10 | GRAB_RAM5 | R | 0h | Value in this field gets loaded from Z1_GRABRAM[11:10] when a read is issued to address location of Z1_GRABRAM in OTP.<br>00 : Invalid. LS5 RAM is inaccessible.<br>01 : Request to allocate LS5 RAM to Zone1.<br>10 : Request to allocate LS5 RAM to Zone1.<br>11 : Request to make LS5 RAM Non-Secure.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-218. Z1_GRABRAMR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 9-8 | GRAB_RAM4 | R | 0h | Value in this field gets loaded from Z1_GRABRAM[9:8] when a read is issued to address location of Z1_GRABRAM in OTP.<br>00 : Invalid. LS4 RAM is inaccessible.<br>01 : Request to allocate LS4 RAM to Zone1.<br>10 : Request to allocate LS4 RAM to Zone1.<br>11 : Request to make LS4 RAM Non-Secure.<br>Reset type: SYSRSn |
| 7-6 | GRAB_RAM3 | R | 0h | Value in this field gets loaded from Z1_GRABRAM[7:6] when a read is issued to address location of Z1_GRABRAM in OTP.<br>00 : Invalid. LS3 RAM is inaccessible.<br>01 : Request to allocate LS3 RAM to Zone1.<br>10 : Request to allocate LS3 RAM to Zone1.<br>11 : Request to make LS3 RAM Non-Secure.<br>Reset type: SYSRSn |
| 5-4 | GRAB_RAM2 | R | 0h | Value in this field gets loaded from Z1_GRABRAM[5:4] when a read is issued to address location of Z1_GRABRAM in OTP.<br>00 : Invalid. LS2 RAM is inaccessible.<br>01 : Request to allocate LS2 RAM to Zone1.<br>10 : Request to allocate LS2 RAM to Zone1.<br>11 : Request to make LS2 RAM Non-Secure.<br>Reset type: SYSRSn |
| 3-2 | GRAB_RAM1 | R | 0h | Value in this field gets loaded from Z1_GRABRAM[3:2] when a read is issued to address location of Z1_GRABRAM in OTP.<br>00 : Invalid. LS1 RAM is inaccessible.<br>01 : Request to allocate LS1 RAM to Zone1.<br>10 : Request to allocate LS1 RAM to Zone1.<br>11 : Request to make LS1 RAM Non-Secure.<br>Reset type: SYSRSn |
| 1-0 | GRAB_RAM0 | R | 0h | Value in this field gets loaded from Z1_GRABRAM[1:0] when a read is issued to address location of Z1_GRABRAM in OTP.<br>00 : Invalid. LS0 RAM is inaccessible.<br>01 : Request to allocate LS0 RAM to Zone1.<br>10 : Request to allocate LS0 RAM to Zone1.<br>11 : Request to make LS0 RAM Non-Secure.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.14.12 Z1_EXEONLYSECTR Register (Offset = 1Eh) [Reset = 00000000h]

Z1_EXEONLYSECTR is shown in Figure 3-198 and described in Table 3-219.

Return to the Summary Table.

Zone 1 Flash Execute_Only Sector Register

**Figure 3-198. Z1_EXEONLYSECTR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | RESERVED | EXEONLY_SECTN | EXEONLY_SECTM | EXEONLY_SECTL | EXEONLY_SECTK | EXEONLY_SECTJ | EXEONLY_SECTI |
| 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_SECTH | EXEONLY_SECTG | EXEONLY_SECTF | EXEONLY_SECTE | EXEONLY_SECTD | EXEONLY_SECTC | EXEONLY_SECTB | EXEONLY_SECTA |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 3-219. Z1_EXEONLYSECTR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15 | RESERVED | R | 0h | Reserved |
| 14 | RESERVED | R | 0h | Reserved |
| 13 | EXEONLY_SECTN | R | 0h | Value in this field gets loaded from Z1_EXEONLYSECT[13:13] when a read is issued to Z1_EXEONLYSECT address location in OTP.<br>0 : Execute-Only protection is enabled for Flash Sector N (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector N (only if it's allocated to Zone1)<br>Reset type: SYSRSn |
| 12 | EXEONLY_SECTM | R | 0h | Value in this field gets loaded from Z1_EXEONLYSECT[12:12] when a read is issued to Z1_EXEONLYSECT address location in OTP.<br>0 : Execute-Only protection is enabled for Flash Sector M (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector M (only if it's allocated to Zone1)<br>Reset type: SYSRSn |
| 11 | EXEONLY_SECTL | R | 0h | Value in this field gets loaded from Z1_EXEONLYSECT[11:11] when a read is issued to Z1_EXEONLYSECT address location in OTP.<br>0 : Execute-Only protection is enabled for Flash Sector L (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector L (only if it's allocated to Zone1)<br>Reset type: SYSRSn |
| 10 | EXEONLY_SECTK | R | 0h | Value in this field gets loaded from Z1_EXEONLYSECT[10:10] when a read is issued to Z1_EXEONLYSECT address location in OTP.<br>0 : Execute-Only protection is enabled for Flash Sector K (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector K (only if it's allocated to Zone1)<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-219. Z1_EXEONLYSECTR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 9 | EXEONLY_SECTJ | R | 0h | Value in this field gets loaded from Z1_EXEONLYSECT[9:9] when a read is issued to Z1_EXEONLYSECT address location in OTP.<br>0 : Execute-Only protection is enabled for Flash Sector J (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector J (only if it's allocated to Zone1)<br>Reset type: SYSRSn |
| 8 | EXEONLY_SECTI | R | 0h | Value in this field gets loaded from Z1_EXEONLYSECT[8:8] when a read is issued to Z1_EXEONLYSECT address location in OTP.<br>0 : Execute-Only protection is enabled for Flash Sector I (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector I (only if it's allocated to Zone1)<br>Reset type: SYSRSn |
| 7 | EXEONLY_SECTH | R | 0h | Value in this field gets loaded from Z1_EXEONLYSECT[7:7] when a read is issued to Z1_EXEONLYSECT address location in OTP.<br>0 : Execute-Only protection is enabled for Flash Sector H (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector H (only if it's allocated to Zone1)<br>Reset type: SYSRSn |
| 6 | EXEONLY_SECTG | R | 0h | Value in this field gets loaded from Z1_EXEONLYSECT[6:6] when a read is issued to Z1_EXEONLYSECT address location in OTP.<br>0 : Execute-Only protection is enabled for Flash Sector G (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector G (only if it's allocated to Zone1)<br>Reset type: SYSRSn |
| 5 | EXEONLY_SECTF | R | 0h | Value in this field gets loaded from Z1_EXEONLYSECT[5:5] when a read is issued to Z1_EXEONLYSECT address location in OTP.<br>0 : Execute-Only protection is enabled for Flash Sector F (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector F (only if it's allocated to Zone1)<br>Reset type: SYSRSn |
| 4 | EXEONLY_SECTE | R | 0h | Value in this field gets loaded from Z1_EXEONLYSECT[4:4] when a read is issued to Z1_EXEONLYSECT address location in OTP.<br>0 : Execute-Only protection is enabled for Flash Sector E (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector E (only if it's allocated to Zone1)<br>Reset type: SYSRSn |
| 3 | EXEONLY_SECTD | R | 0h | Value in this field gets loaded from Z1_EXEONLYSECT[3:3] when a read is issued to Z1_EXEONLYSECT address location in OTP.<br>0 : Execute-Only protection is enabled for Flash Sector D (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector D (only if it's allocated to Zone1)<br>Reset type: SYSRSn |
| 2 | EXEONLY_SECTC | R | 0h | Value in this field gets loaded from Z1_EXEONLYSECT[2:2] when a read is issued to Z1_EXEONLYSECT address location in OTP.<br>0 : Execute-Only protection is enabled for Flash Sector C (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector C (only if it's allocated to Zone1)<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-219. Z1_EXEONLYSECTR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1 | EXEONLY_SECTB | R | 0h | Value in this field gets loaded from Z1_EXEONLYSECT[1:1] when a read is issued to Z1_EXEONLYSECT address location in OTP.<br>0 : Execute-Only protection is enabled for Flash Sector B (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector B (only if it's allocated to Zone1)<br>Reset type: SYSRSn |
| 0 | EXEONLY_SECTA | R | 0h | Value in this field gets loaded from Z1_EXEONLYSECT[0:0] when a read is issued to Z1_EXEONLYSECT address location in OTP.<br>0 : Execute-Only protection is enabled for Flash Sector A (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector A (only if it's allocated to Zone1)<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.14.13 Z1_EXEONLYRAMR Register (Offset = 20h) [Reset = 00000000h]

Z1_EXEONLYRAMR is shown in Figure 3-199 and described in Table 3-220.

Return to the Summary Table.

Zone 1 RAM Execute_Only Block Register

**Figure 3-199. Z1_EXEONLYRAMR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EXEONLY_RAM7 | EXEONLY_RAM6 | EXEONLY_RAM5 | EXEONLY_RAM4 | EXEONLY_RAM3 | EXEONLY_RAM2 | EXEONLY_RAM1 | EXEONLY_RAM0 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 3-220. Z1_EXEONLYRAMR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-8 | RESERVED | R | 0h | Reserved |
| 7 | EXEONLY_RAM7 | R | 0h | Value in this field gets loaded from Z1_EXEONLYRAM[7:7] when a read is issued to Z1_EXEONLYRAM address location in OTP.<br>0 : Execute-Only protection is enabled for D1 RAM (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for D1 RAM (only if it's allocated to Zone1)<br>Reset type: SYSRSn |
| 6 | EXEONLY_RAM6 | R | 0h | Value in this field gets loaded from Z1_EXEONLYRAM[6:6] when a read is issued to Z1_EXEONLYRAM address location in OTP.<br>0 : Execute-Only protection is enabled for D0 RAM (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for D0 RAM (only if it's allocated to Zone1)<br>Reset type: SYSRSn |
| 5 | EXEONLY_RAM5 | R | 0h | Value in this field gets loaded from Z1_EXEONLYRAM[5:5] when a read is issued to Z1_EXEONLYRAM address location in OTP.<br>0 : Execute-Only protection is enabled for LS5 RAM (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for LS5 RAM (only if it's allocated to Zone1)<br>Reset type: SYSRSn |
| 4 | EXEONLY_RAM4 | R | 0h | Value in this field gets loaded from Z1_EXEONLYRAM[4:4] when a read is issued to Z1_EXEONLYRAM address location in OTP.<br>0 : Execute-Only protection is enabled for LS4 RAM (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for LS4 RAM (only if it's allocated to Zone1)<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-220. Z1_EXEONLYRAMR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3 | EXEONLY_RAM3 | R | 0h | Value in this field gets loaded from Z1_EXEONLYRAM[3:3] when a read is issued to Z1_EXEONLYRAM address location in OTP.<br>0 : Execute-Only protection is enabled for LS3 RAM (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for LS3 RAM (only if it's allocated to Zone1)<br>Reset type: SYSRSn |
| 2 | EXEONLY_RAM2 | R | 0h | Value in this field gets loaded from Z1_EXEONLYRAM[2:2] when a read is issued to Z1_EXEONLYRAM address location in OTP.<br>0 : Execute-Only protection is enabled for LS2 RAM (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for LS2 RAM (only if it's allocated to Zone1)<br>Reset type: SYSRSn |
| 1 | EXEONLY_RAM1 | R | 0h | Value in this field gets loaded from Z1_EXEONLYRAM[1:1] when a read is issued to Z1_EXEONLYRAM address location in OTP.<br>0 : Execute-Only protection is enabled for LS1 RAM (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for LS1 RAM (only if it's allocated to Zone1)<br>Reset type: SYSRSn |
| 0 | EXEONLY_RAM0 | R | 0h | Value in this field gets loaded from Z1_EXEONLYRAM[0:0] when a read is issued to Z1_EXEONLYRAM address location in OTP.<br>0 : Execute-Only protection is enabled for LS0 RAM (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for LS0 RAM (only if it's allocated to Zone1)<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.15 DCSM_Z2_REGS Registers

Table 3-221 lists the memory-mapped registers for the DCSM_Z2_REGS registers. All register offset addresses not listed in Table 3-221 should be considered as reserved locations and the register contents should not be modified.

**Table 3-221. DCSM_Z2_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | Z2_LINKPOINTER | Zone 2 Link Pointer | | Go |
| 2h | Z2_OTPSECLOCK | Zone 2 OTP Secure JTAG lock | | Go |
| 4h | Z2_BOOTCTRL | Boot Mode | | Go |
| 6h | Z2_LINKPOINTERERR | Link Pointer Error | | Go |
| 10h | Z2_CSMKEY0 | Zone 2 CSM Key 0 | | Go |
| 12h | Z2_CSMKEY1 | Zone 2 CSM Key 1 | | Go |
| 14h | Z2_CSMKEY2 | Zone 2 CSM Key 2 | | Go |
| 16h | Z2_CSMKEY3 | Zone 2 CSM Key 3 | | Go |
| 19h | Z2_CR | Zone 2 CSM Control Register | | Go |
| 1Ah | Z2_GRABSECTR | Zone 2 Grab Flash Sectors Register | | Go |
| 1Ch | Z2_GRABRAMR | Zone 2 Grab RAM Blocks Register | | Go |
| 1Eh | Z2_EXEONLYSECTR | Zone 2 Flash Execute_Only Sector Register | | Go |
| 20h | Z2_EXEONLYRAMR | Zone 2 RAM Execute_Only Block Register | | Go |

Complex bit access types are encoded to fit into small table cells. Table 3-222 shows the codes that are used for access types in this section.

**Table 3-222. DCSM_Z2_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| R-0 | R<br>-0 | Read<br>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. |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.15.1 Z2_LINKPOINTER Register (Offset = 0h) [Reset = E0000000h]

Z2_LINKPOINTER is shown in Figure 3-200 and described in Table 3-223.

Return to the Summary Table.

Zone 2 Link Pointer

**Figure 3-200. Z2_LINKPOINTER Register**

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

**Table 3-223. Z2_LINKPOINTER Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-29 | RESERVED | R | 7h | Reserved |
| 28-0 | LINKPOINTER | R | 0h | This is the Resolved Link-Pointer value which is generated by looking at the three physical Link-Pointer values loaded from OTP.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.15.2 Z2_OTPSECLOCK Register (Offset = 2h) [Reset = 00000FFFh]

Z2_OTPSECLOCK is shown in Figure 3-201 and described in Table 3-224.

Return to the Summary Table.

Zone 2 OTP Secure JTAG lock

**Figure 3-201. Z2_OTPSECLOCK Register**

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

**Table 3-224. Z2_OTPSECLOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-12 | RESERVED | R | 0h | Reserved |
| 11-8 | CRCLOCK | R | Fh | Value in this field gets loaded from Z2_CRCLOCK[3:0] when a read is issued to address location of Z2_CRCLOCK in OTP.<br>1111 : VCU has ability to calculate CRC on secure memories.<br>Other Value : VCU doesn't have ability to calculate CRC on secure memories.<br>Reset type: SYSRSn |
| 7-4 | PSWDLOCK | R | Fh | Value in this field gets loaded from Z2_PSWDLOCK[3:0] when a read is issued to address location of Z2_PSWDLOCK in OTP.<br>1111 : CSM password locations in OTP are not protected and can be read from debugger as well as code running from anywhere.<br>Other Value : CSM password locations in OTP are protected and can't be read without unlocking CSM of that zone.<br>Reset type: SYSRSn |
| 3-0 | JTAGLOCK | R | Fh | Value in this field gets loaded from Z2_JTAGLOCK[3:0] when a read is issued to address location of Z2_JTAGLOCK in OTP.<br>1111 : JTAG/Emulation access is allowed.<br>Other Value : JTAG/Emulation access not allowed.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.15.3 Z2_BOOTCTRL Register (Offset = 4h) [Reset = 00000000h]

Z2_BOOTCTRL is shown in Figure 3-202 and described in Table 3-225.

Return to the Summary Table.

Boot Mode

**Figure 3-202. Z2_BOOTCTRL Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BOOTPIN1 | | | | | | | | BOOTPIN0 | | | | | | | | BMODE | | | | | | | | KEY | | | | | | | |
| R-0h | | | | | | | | R-0h | | | | | | | | R-0h | | | | | | | | R-0h | | | | | | | |

**Table 3-225. Z2_BOOTCTRL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-24 | BOOTPIN1 | R | 0h | This field gets loaded with Z2_BOOTCTRL[31:24] when a dummy read is issued to address location of Z2_BOOTCTRL in OTP.<br>This assigns the pin to be used as BOOTPIN1.<br>0 : Pick default bootmode pin.<br>1 : Pick GPIO0 as BOOTPIN1.<br>2 : Pick GPIO1 as BOOTPIN1.<br>....<br>....<br>n : Pick GPIOn-1 as BOOTPIN1.<br>Reset type: SYSRSn |
| 23-16 | BOOTPIN0 | R | 0h | This field gets loaded with Z2_BOOTCTRL[23:16] when a dummy read is issued to address location of Z2_BOOTCTRL in OTP.<br>This assigns the pin to be used as BOOTPIN1.<br>0 : Pick default bootmode pin.<br>1 : Pick GPIO0 as BOOTPIN1.<br>2 : Pick GPIO1 as BOOTPIN1.<br>....<br>....<br>n : Pick GPIOn-1 as BOOTPIN1.<br>Reset type: SYSRSn |
| 15-8 | BMODE | R | 0h | This field gets loaded with Z2_BOOTCTRL[15:8] when a dummy read is issued to address location of Z2_BOOTCTRL in OTP.<br>Reset type: SYSRSn |
| 7-0 | KEY | R | 0h | This field gets loaded with Z2_BOOTCTRL[7:0] when a dummy read is issued to address location of Z2_BOOTCTRL in OTP.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.15.4 Z2_LINKPOINTERERR Register (Offset = 6h) [Reset = FFFFFFFFh]

Z2_LINKPOINTERERR is shown in Figure 3-203 and described in Table 3-226.

Return to the Summary Table.

Link Pointer Error

**Figure 3-203. Z2_LINKPOINTERERR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Z2_LINKPOINTERERR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R-FFFFFFFFh | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 3-226. Z2_LINKPOINTERERR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | Z2_LINKPOINTERERR | R | FFFFFFFFh | These bits indicate errors during formation of the resolved Link-Pointer value after the three physical Link-Pointer values loaded of OTP.<br>0 : No Error.<br>Other : Error on bit positions which is set to 1.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.15.5 Z2_CSMKEY0 Register (Offset = 10h) [Reset = 00000000h]

Z2_CSMKEY0 is shown in Figure 3-204 and described in Table 3-227.

Return to the Summary Table.

Zone 2 CSM Key 0

**Figure 3-204. Z2_CSMKEY0 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Z2_CSMKEY0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 3-227. Z2_CSMKEY0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | Z2_CSMKEY0 | R | 0h | To unclock Zone2, user needs to write this regsiter 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.)<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.15.6 Z2_CSMKEY1 Register (Offset = 12h) [Reset = 00000000h]

Z2_CSMKEY1 is shown in Figure 3-205 and described in Table 3-228.

Return to the Summary Table.

Zone 2 CSM Key 1

**Figure 3-205. Z2_CSMKEY1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Z2_CSMKEY1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 3-228. Z2_CSMKEY1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | Z2_CSMKEY1 | R | 0h | To unclock Zone2, user needs to write this regsiter 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.)<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.15.7 Z2_CSMKEY2 Register (Offset = 14h) [Reset = 00000000h]

Z2_CSMKEY2 is shown in Figure 3-206 and described in Table 3-229.

Return to the Summary Table.

Zone 2 CSM Key 2

**Figure 3-206. Z2_CSMKEY2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Z2_CSMKEY2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 3-229. Z2_CSMKEY2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | Z2_CSMKEY2 | R | 0h | To unclock Zone2, user needs to write this regsiter 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.)<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.15.8 Z2_CSMKEY3 Register (Offset = 16h) [Reset = 00000000h]

Z2_CSMKEY3 is shown in Figure 3-207 and described in Table 3-230.

Return to the Summary Table.

Zone 2 CSM Key 3

**Figure 3-207. Z2_CSMKEY3 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Z2_CSMKEY3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 3-230. Z2_CSMKEY3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | Z2_CSMKEY3 | R | 0h | To unclock Zone2, user needs to write this regsiter 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.)<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.15.9 Z2_CR Register (Offset = 19h) [Reset = 0008h]

Z2_CR is shown in Figure 3-208 and described in Table 3-231.

Return to the Summary Table.

Zone 2 CSM Control Register

**Figure 3-208. Z2_CR Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| FORCESEC | RESERVED | | | | | | |
| R-0/W-0h | R-0-0h | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | ARMED | UNSECURE | ALLONE | ALLZERO | RESERVED | | |
| R-0h | R-0h | R-0h | R-0h | R-1h | R-0h | | |

**Table 3-231. Z2_CR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | FORCESEC | R-0/W | 0h | A write '1' to this fields resets the state of zone. If zone is unlocked, it'll lock(secure) the zone and also resets all the bits in this register.<br>Reset type: SYSRSn |
| 14-8 | RESERVED | R-0 | 0h | Reserved |
| 7 | RESERVED | R | 0h | Reserved |
| 6 | ARMED | R | 0h | 0 : Dummy read to CSM Password locations in OTP hasn't been performed.<br>1 : Dummy read to CSM Password locations in OTP has been performed.<br>Reset type: SYSRSn |
| 5 | UNSECURE | R | 0h | Indiacates the state of Zone.<br>0 : Zone is in lock(secure) state.<br>1 : Zone is in unlock(unsecure) state.<br>Reset type: SYSRSn |
| 4 | ALLONE | R | 0h | Indicates the state of CSM passowrds.<br>0 : CSM Passwords are not all ones.<br>1 : CSM Passwords are all ones and zone is in unlock state.<br>Reset type: SYSRSn |
| 3 | ALLZERO | R | 1h | Indicates the state of CSM passowrds.<br>0 : CSM Passwords are not all zeros.<br>1 : CSM Passwords are all zero and device is permanently locked.<br>Reset type: SYSRSn |
| 2-0 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.15.10 Z2_GRABSECTR Register (Offset = 1Ah) [Reset = 00000000h]

Z2_GRABSECTR is shown in Figure 3-209 and described in Table 3-232.

Return to the Summary Table.

Zone 2 Grab Flash Sectors Register

**Figure 3-209. Z2_GRABSECTR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | RESERVED | | GRAB_SECTN | | GRAB_SECTM | |
| R-0h | | R-0h | | R-0h | | R-0h | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GRAB_SECTL | | GRAB_SECTK | | GRAB_SECTJ | | GRAB_SECTI | |
| R-0h | | R-0h | | R-0h | | R-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GRAB_SECTH | | GRAB_SECTG | | GRAB_SECTF | | GRAB_SECTE | |
| R-0h | | R-0h | | R-0h | | R-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GRAB_SECTD | | GRAB_SECTC | | GRAB_SECTB | | GRAB_SECTA | |
| R-0h | | R-0h | | R-0h | | R-0h | |

**Table 3-232. Z2_GRABSECTR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | RESERVED | R | 0h | Reserved |
| 29-28 | RESERVED | R | 0h | Reserved |
| 27-26 | GRAB_SECTN | R | 0h | Value in this field gets loaded from Z2_GRABSECT[27:26] when a read is issued to address location of Z2_GRABSECT in OTP.<br>00 : Invalid. Flash Sector N is inaccessible.<br>01 : Request to allocate Flash Sector N to Zone2.<br>10 : Request to allocate Flash Sector N to Zone2.<br>11 : Request to make Flash sector N Non-Secure.<br>Reset type: SYSRSn |
| 25-24 | GRAB_SECTM | R | 0h | Value in this field gets loaded from Z2_GRABSECT[25:24] when a read is issued to address location of Z2_GRABSECT in OTP.<br>00 : Invalid. Flash Sector M is inaccessible.<br>01 : Request to allocate Flash Sector M to Zone2.<br>10 : Request to allocate Flash Sector M to Zone2.<br>11 : Request to make Flash sector M Non-Secure.<br>Reset type: SYSRSn |
| 23-22 | GRAB_SECTL | R | 0h | Value in this field gets loaded from Z2_GRABSECT[23:22] when a read is issued to address location of Z2_GRABSECT in OTP.<br>00 : Invalid. Flash Sector L is inaccessible.<br>01 : Request to allocate Flash Sector L to Zone2.<br>10 : Request to allocate Flash Sector L to Zone2.<br>11 : Request to make Flash sector L Non-Secure.<br>Reset type: SYSRSn |
| 21-20 | GRAB_SECTK | R | 0h | Value in this field gets loaded from Z2_GRABSECT[21:20] when a read is issued to address location of Z2_GRABSECT in OTP.<br>00 : Invalid. Flash Sector K is inaccessible.<br>01 : Request to allocate Flash Sector K to Zone2.<br>10 : Request to allocate Flash Sector K to Zone2.<br>11 : Request to make Flash sector K Non-Secure.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-232. Z2_GRABSECTR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19-18 | GRAB_SECTJ | R | 0h | Value in this field gets loaded from Z2_GRABSECT[19:18] when a read is issued to address location of Z2_GRABSECT in OTP.<br>00 : Invalid. Flash Sector J is inaccessible.<br>01 : Request to allocate Flash Sector J to Zone2.<br>10 : Request to allocate Flash Sector J to Zone2.<br>11 : Request to make Flash sector J Non-Secure.<br>Reset type: SYSRSn |
| 17-16 | GRAB_SECTI | R | 0h | Value in this field gets loaded from Z2_GRABSECT[17:16] when a read is issued to address location of Z2_GRABSECT in OTP.<br>00 : Invalid. Flash Sector I is inaccessible.<br>01 : Request to allocate Flash Sector I to Zone2.<br>10 : Request to allocate Flash Sector I to Zone2.<br>11 : Request to make Flash sector I Non-Secure.<br>Reset type: SYSRSn |
| 15-14 | GRAB_SECTH | R | 0h | Value in this field gets loaded from Z2_GRABSECT[15:14] when a read is issued to address location of Z2_GRABSECT in OTP.<br>00 : Invalid. Flash Sector H is inaccessible.<br>01 : Request to allocate Flash Sector H to Zone2.<br>10 : Request to allocate Flash Sector H to Zone2.<br>11 : Request to make Flash sector H Non-Secure.<br>Reset type: SYSRSn |
| 13-12 | GRAB_SECTG | R | 0h | Value in this field gets loaded from Z2_GRABSECT[13:12] when a read is issued to address location of Z2_GRABSECT in OTP.<br>00 : Invalid. Flash Sector G is inaccessible.<br>01 : Request to allocate Flash Sector G to Zone2.<br>10 : Request to allocate Flash Sector G to Zone2.<br>11 : Request to make Flash sector G Non-Secure.<br>Reset type: SYSRSn |
| 11-10 | GRAB_SECTF | R | 0h | Value in this field gets loaded from Z2_GRABSECT[11:10] when a read is issued to address location of Z2_GRABSECT in OTP.<br>00 : Invalid. Flash Sector F is inaccessible.<br>01 : Request to allocate Flash Sector F to Zone2.<br>10 : Request to allocate Flash Sector F to Zone2.<br>11 : Request to make Flash sector F Non-Secure.<br>Reset type: SYSRSn |
| 9-8 | GRAB_SECTE | R | 0h | Value in this field gets loaded from Z2_GRABSECT[9:8] when a read is issued to address location of Z2_GRABSECT in OTP.<br>00 : Invalid. Flash Sector E is inaccessible.<br>01 : Request to allocate Flash Sector E to Zone2.<br>10 : Request to allocate Flash Sector E to Zone2.<br>11 : Request to make Flash sector E Non-Secure.<br>Reset type: SYSRSn |
| 7-6 | GRAB_SECTD | R | 0h | Value in this field gets loaded from Z2_GRABSECT[7:6] when a read is issued to address location of Z2_GRABSECT in OTP.<br>00 : Invalid. Flash Sector D is inaccessible.<br>01 : Request to allocate Flash Sector D to Zone2.<br>10 : Request to allocate Flash Sector D to Zone2.<br>11 : Request to make Flash sector D Non-Secure.<br>Reset type: SYSRSn |
| 5-4 | GRAB_SECTC | R | 0h | Value in this field gets loaded from Z2_GRABSECT[5:4] when a read is issued to address location of Z2_GRABSECT in OTP.<br>00 : Invalid. Flash Sector C is inaccessible.<br>01 : Request to allocate Flash Sector C to Zone2.<br>10 : Request to allocate Flash Sector C to Zone2.<br>11 : Request to make Flash sector C Non-Secure.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-232. Z2_GRABSECTR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-2 | GRAB_SECTB | R | 0h | Value in this field gets loaded from Z2_GRABSECT[3:2] when a read is issued to address location of Z2_GRABSECT in OTP.<br>00 : Invalid. Flash Sector B is inaccessible.<br>01 : Request to allocate Flash Sector B to Zone2.<br>10 : Request to allocate Flash Sector B to Zone2.<br>11 : Request to make Flash sector B Non-Secure.<br>Reset type: SYSRSn |
| 1-0 | GRAB_SECTA | R | 0h | Value in this field gets loaded from Z2_GRABSECT[1:0] when a read is issued to address location of Z2_GRABSECT in OTP.<br>00 : Invalid. Flash Sector A is inaccessible.<br>01 : Request to allocate Flash Sector A to Zone2.<br>10 : Request to allocate Flash Sector A to Zone2.<br>11 : Request to make Flash sector A Non-Secure.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.15.11 Z2_GRABRAMR Register (Offset = 1Ch) [Reset = 00000000h]

Z2_GRABRAMR is shown in Figure 3-210 and described in Table 3-233.

Return to the Summary Table.

Zone 2 Grab RAM Blocks Register

**Figure 3-210. Z2_GRABRAMR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | GRAB_CLA1 | | RESERVED | | | |
| R-0h | | R-0h | | R-0h | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GRAB_RAM7 | | GRAB_RAM6 | | GRAB_RAM5 | | GRAB_RAM4 | |
| R-0h | | R-0h | | R-0h | | R-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GRAB_RAM3 | | GRAB_RAM2 | | GRAB_RAM1 | | GRAB_RAM0 | |
| R-0h | | R-0h | | R-0h | | R-0h | |

**Table 3-233. Z2_GRABRAMR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | RESERVED | R | 0h | Reserved |
| 29-28 | GRAB_CLA1 | R | 0h | Value in this field gets loaded from Z2_GRABRAM[29:28] when a read is issued to address location of Z2_GRABRAM in OTP.<br>00 : Invalid. CLA1 is inaccessible.<br>01 : Request to allocate CLA1 to Zone2.<br>10 : Request to allocate CLA1 to Zone2.<br>11 : Request to make CLA1 Non-Secure.<br>Reset type: SYSRSn |
| 27-16 | RESERVED | R | 0h | Reserved |
| 15-14 | GRAB_RAM7 | R | 0h | Value in this field gets loaded from Z2_GRABRAM[15:14] when a read is issued to address location of Z2_GRABRAM in OTP.<br>00 : Invalid. D1 RAM is inaccessible.<br>01 : Request to allocate D1 RAM to Zone2.<br>10 : Request to allocate D1 RAM to Zone2.<br>11 : Request to make D1 RAM Non-Secure.<br>Reset type: SYSRSn |
| 13-12 | GRAB_RAM6 | R | 0h | Value in this field gets loaded from Z2_GRABRAM[13:12] when a read is issued to address location of Z2_GRABRAM in OTP.<br>00 : Invalid. D0 RAM is inaccessible.<br>01 : Request to allocate D0 RAM to Zone2.<br>10 : Request to allocate D0 RAM to Zone2.<br>11 : Request to make D0 RAM Non-Secure.<br>Reset type: SYSRSn |
| 11-10 | GRAB_RAM5 | R | 0h | Value in this field gets loaded from Z2_GRABRAM[11:10] when a read is issued to address location of Z2_GRABRAM in OTP.<br>00 : Invalid. LS5 RAM is inaccessible.<br>01 : Request to allocate LS5 RAM to Zone2.<br>10 : Request to allocate LS5 RAM to Zone2.<br>11 : Request to make LS5 RAM Non-Secure.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-233. Z2_GRABRAMR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 9-8 | GRAB_RAM4 | R | 0h | Value in this field gets loaded from Z2_GRABRAM[9:8] when a read is issued to address location of Z2_GRABRAM in OTP.<br>00 : Invalid. LS4 RAM is inaccessible.<br>01 : Request to allocate LS4 RAM to Zone2.<br>10 : Request to allocate LS4 RAM to Zone2.<br>11 : Request to make LS4 RAM Non-Secure.<br>Reset type: SYSRSn |
| 7-6 | GRAB_RAM3 | R | 0h | Value in this field gets loaded from Z2_GRABRAM[7:6] when a read is issued to address location of Z2_GRABRAM in OTP.<br>00 : Invalid. LS3 RAM is inaccessible.<br>01 : Request to allocate LS3 RAM to Zone2.<br>10 : Request to allocate LS3 RAM to Zone2.<br>11 : Request to make LS3 RAM Non-Secure.<br>Reset type: SYSRSn |
| 5-4 | GRAB_RAM2 | R | 0h | Value in this field gets loaded from Z2_GRABRAM[5:4] when a read is issued to address location of Z2_GRABRAM in OTP.<br>00 : Invalid. LS2 RAM is inaccessible.<br>01 : Request to allocate LS2 RAM to Zone2.<br>10 : Request to allocate LS2 RAM to Zone2.<br>11 : Request to make LS2 RAM Non-Secure.<br>Reset type: SYSRSn |
| 3-2 | GRAB_RAM1 | R | 0h | Value in this field gets loaded from Z2_GRABRAM[3:2] when a read is issued to address location of Z2_GRABRAM in OTP.<br>00 : Invalid. LS1 RAM is inaccessible.<br>01 : Request to allocate LS1 RAM to Zone2.<br>10 : Request to allocate LS1 RAM to Zone2.<br>11 : Request to make LS1 RAM Non-Secure.<br>Reset type: SYSRSn |
| 1-0 | GRAB_RAM0 | R | 0h | Value in this field gets loaded from Z2_GRABRAM[1:0] when a read is issued to address location of Z2_GRABRAM in OTP.<br>00 : Invalid. LS0 RAM is inaccessible.<br>01 : Request to allocate LS0 RAM to Zone2.<br>10 : Request to allocate LS0 RAM to Zone2.<br>11 : Request to make LS0 RAM Non-Secure.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.15.12 Z2_EXEONLYSECTR Register (Offset = 1Eh) [Reset = 00000000h]

Z2_EXEONLYSECTR is shown in Figure 3-211 and described in Table 3-234.

Return to the Summary Table.

Zone 2 Flash Execute_Only Sector Register

**Figure 3-211. Z2_EXEONLYSECTR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | RESERVED | EXEONLY_SECTN | EXEONLY_SECTM | EXEONLY_SECTL | EXEONLY_SECTK | EXEONLY_SECTJ | EXEONLY_SECTI |
| 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_SECTH | EXEONLY_SECTG | EXEONLY_SECTF | EXEONLY_SECTE | EXEONLY_SECTD | EXEONLY_SECTC | EXEONLY_SECTB | EXEONLY_SECTA |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 3-234. Z2_EXEONLYSECTR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15 | RESERVED | R | 0h | Reserved |
| 14 | RESERVED | R | 0h | Reserved |
| 13 | EXEONLY_SECTN | R | 0h | Value in this field gets loaded from Z2_EXEONLYSECT[13:13] when a read is issued to Z2_EXEONLYSECT address location in OTP.<br>0 : Execute-Only protection is enabled for Flash Sector N (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector N (only if it's allocated to Zone2)<br>Reset type: SYSRSn |
| 12 | EXEONLY_SECTM | R | 0h | Value in this field gets loaded from Z2_EXEONLYSECT[12:12] when a read is issued to Z2_EXEONLYSECT address location in OTP.<br>0 : Execute-Only protection is enabled for Flash Sector M (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector M (only if it's allocated to Zone2)<br>Reset type: SYSRSn |
| 11 | EXEONLY_SECTL | R | 0h | Value in this field gets loaded from Z2_EXEONLYSECT[11:11] when a read is issued to Z2_EXEONLYSECT address location in OTP.<br>0 : Execute-Only protection is enabled for Flash Sector L (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector L (only if it's allocated to Zone2)<br>Reset type: SYSRSn |
| 10 | EXEONLY_SECTK | R | 0h | Value in this field gets loaded from Z2_EXEONLYSECT[10:10] when a read is issued to Z2_EXEONLYSECT address location in OTP.<br>0 : Execute-Only protection is enabled for Flash Sector K (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector K (only if it's allocated to Zone2)<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-234. Z2_EXEONLYSECTR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 9 | EXEONLY_SECTJ | R | 0h | Value in this field gets loaded from Z2_EXEONLYSECT[9:9] when a read is issued to Z2_EXEONLYSECT address location in OTP.<br>0 : Execute-Only protection is enabled for Flash Sector J (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector J (only if it's allocated to Zone2)<br>Reset type: SYSRSn |
| 8 | EXEONLY_SECTI | R | 0h | Value in this field gets loaded from Z2_EXEONLYSECT[8:8] when a read is issued to Z2_EXEONLYSECT address location in OTP.<br>0 : Execute-Only protection is enabled for Flash Sector I (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector I (only if it's allocated to Zone2)<br>Reset type: SYSRSn |
| 7 | EXEONLY_SECTH | R | 0h | Value in this field gets loaded from Z2_EXEONLYSECT[7:7] when a read is issued to Z2_EXEONLYSECT address location in OTP.<br>0 : Execute-Only protection is enabled for Flash Sector H (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector H (only if it's allocated to Zone2)<br>Reset type: SYSRSn |
| 6 | EXEONLY_SECTG | R | 0h | Value in this field gets loaded from Z2_EXEONLYSECT[6:6] when a read is issued to Z2_EXEONLYSECT address location in OTP.<br>0 : Execute-Only protection is enabled for Flash Sector G (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector G (only if it's allocated to Zone2)<br>Reset type: SYSRSn |
| 5 | EXEONLY_SECTF | R | 0h | Value in this field gets loaded from Z2_EXEONLYSECT[5:5] when a read is issued to Z2_EXEONLYSECT address location in OTP.<br>0 : Execute-Only protection is enabled for Flash Sector F (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector F (only if it's allocated to Zone2)<br>Reset type: SYSRSn |
| 4 | EXEONLY_SECTE | R | 0h | Value in this field gets loaded from Z2_EXEONLYSECT[4:4] when a read is issued to Z2_EXEONLYSECT address location in OTP.<br>0 : Execute-Only protection is enabled for Flash Sector E (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector E (only if it's allocated to Zone2)<br>Reset type: SYSRSn |
| 3 | EXEONLY_SECTD | R | 0h | Value in this field gets loaded from Z2_EXEONLYSECT[3:3] when a read is issued to Z2_EXEONLYSECT address location in OTP.<br>0 : Execute-Only protection is enabled for Flash Sector D (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector D (only if it's allocated to Zone2)<br>Reset type: SYSRSn |
| 2 | EXEONLY_SECTC | R | 0h | Value in this field gets loaded from Z2_EXEONLYSECT[2:2] when a read is issued to Z2_EXEONLYSECT address location in OTP.<br>0 : Execute-Only protection is enabled for Flash Sector C (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector C (only if it's allocated to Zone2)<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-234. Z2_EXEONLYSECTR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1 | EXEONLY_SECTB | R | 0h | Value in this field gets loaded from Z2_EXEONLYSECT[1:1] when a read is issued to Z2_EXEONLYSECT address location in OTP.<br>0 : Execute-Only protection is enabled for Flash Sector B (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector B (only if it's allocated to Zone2)<br>Reset type: SYSRSn |
| 0 | EXEONLY_SECTA | R | 0h | Value in this field gets loaded from Z2_EXEONLYSECT[0:0] when a read is issued to Z2_EXEONLYSECT address location in OTP.<br>0 : Execute-Only protection is enabled for Flash Sector A (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector A (only if it's allocated to Zone2)<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.15.13 Z2_EXEONLYRAMR Register (Offset = 20h) [Reset = 00000000h]

Z2_EXEONLYRAMR is shown in Figure 3-212 and described in Table 3-235.

Return to the Summary Table.

Zone 2 RAM Execute_Only Block Register

**Figure 3-212. Z2_EXEONLYRAMR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EXEONLY_RAM7 | EXEONLY_RAM6 | EXEONLY_RAM5 | EXEONLY_RAM4 | EXEONLY_RAM3 | EXEONLY_RAM2 | EXEONLY_RAM1 | EXEONLY_RAM0 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 3-235. Z2_EXEONLYRAMR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-8 | RESERVED | R-0 | 0h | Reserved |
| 7 | EXEONLY_RAM7 | R | 0h | Value in this field gets loaded from Z2_EXEONLYRAM[7:7] when a read is issued to Z2_EXEONLYRAM address location in OTP.<br>0 : Execute-Only protection is enabled for D1 RAM (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for D1 RAM (only if it's allocated to Zone2)<br>Reset type: SYSRSn |
| 6 | EXEONLY_RAM6 | R | 0h | Value in this field gets loaded from Z2_EXEONLYRAM[6:6] when a read is issued to Z2_EXEONLYRAM address location in OTP.<br>0 : Execute-Only protection is enabled for D0 RAM (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for D0 RAM (only if it's allocated to Zone2)<br>Reset type: SYSRSn |
| 5 | EXEONLY_RAM5 | R | 0h | Value in this field gets loaded from Z2_EXEONLYRAM[5:5] when a read is issued to Z2_EXEONLYRAM address location in OTP.<br>0 : Execute-Only protection is enabled for LS5 RAM (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for LS5 RAM (only if it's allocated to Zone2)<br>Reset type: SYSRSn |
| 4 | EXEONLY_RAM4 | R | 0h | Value in this field gets loaded from Z2_EXEONLYRAM[4:4] when a read is issued to Z2_EXEONLYRAM address location in OTP.<br>0 : Execute-Only protection is enabled for LS4 RAM (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for LS4 RAM (only if it's allocated to Zone2)<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-235. Z2_EXEONLYRAMR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3 | EXEONLY_RAM3 | R | 0h | Value in this field gets loaded from Z2_EXEONLYRAM[3:3] when a read is issued to Z2_EXEONLYRAM address location in OTP.<br>0 : Execute-Only protection is enabled for LS3 RAM (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for LS3 RAM (only if it's allocated to Zone2)<br>Reset type: SYSRSn |
| 2 | EXEONLY_RAM2 | R | 0h | Value in this field gets loaded from Z2_EXEONLYRAM[2:2] when a read is issued to Z2_EXEONLYRAM address location in OTP.<br>0 : Execute-Only protection is enabled for LS2 RAM (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for LS2 RAM (only if it's allocated to Zone2)<br>Reset type: SYSRSn |
| 1 | EXEONLY_RAM1 | R | 0h | Value in this field gets loaded from Z2_EXEONLYRAM[1:1] when a read is issued to Z2_EXEONLYRAM address location in OTP.<br>0 : Execute-Only protection is enabled for LS1 RAM (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for LS1 RAM (only if it's allocated to Zone2)<br>Reset type: SYSRSn |
| 0 | EXEONLY_RAM0 | R | 0h | Value in this field gets loaded from Z2_EXEONLYRAM[0:0] when a read is issued to Z2_EXEONLYRAM address location in OTP.<br>0 : Execute-Only protection is enabled for LS0 RAM (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for LS0 RAM (only if it's allocated to Zone2)<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.16 DCSM_COMMON_REGS Registers

Table 3-236 lists the memory-mapped registers for the DCSM_COMMON_REGS registers. All register offset addresses not listed in Table 3-236 should be considered as reserved locations and the register contents should not be modified.

**Table 3-236. DCSM_COMMON_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | FLSEM | Flash Wrapper Semaphore Register | EALLOW | Go |
| 2h | SECTSTAT | Sectors Status Register | | Go |
| 4h | RAMSTAT | RAM Status Register | | Go |

Complex bit access types are encoded to fit into small table cells. Table 3-237 shows the codes that are used for access types in this section.

**Table 3-237. DCSM_COMMON_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| R-0 | R<br>-0 | Read<br>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. |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.16.1 FLSEM Register (Offset = 0h) [Reset = 00000000h]

FLSEM is shown in Figure 3-213 and described in Table 3-238.

Return to the Summary Table.

Flash Wrapper Semaphore Register

**Figure 3-213. FLSEM Register**

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

**Table 3-238. 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.<br>Reset type: SYSRSn |
| 7-2 | RESERVED | R | 0h | Reserved |
| 1-0 | SEM | R/W | 0h | 00 : C28X Flash Wrapper registers can be written by code running from non-secure zone.<br>01 : Flash Wrapper registers can be written by code running from Zone1 security zone only. User must set this value to perform flash operation on flash sectors of Zone1.<br>10 : C28X Flash Wrapper registers can be written by code running from Zone2 security zone only. User must set this value to perform flash operation on flash sectors of Zone2.<br>11 : C28X Flash Wrapper registers can be written by code running from non-secure zone.<br>Allowed State Transitions in this field.<br>00 to 11 : Code running from anywhere.<br>11 to 00 : Not allowed.<br>00/11 to 01 : Code running from Zone1 only can perform this transition.<br>01 to 00/11 : Code running from Zone1 only can perform this transition.<br>00/11 to 10 : Code running from Zone2 only can perform this transition.<br>10 to 00/11 : Code running from Zone2 can perform this transition<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.16.2 SECTSTAT Register (Offset = 2h) [Reset = 00000000h]

SECTSTAT is shown in Figure 3-214 and described in Table 3-239.

Return to the Summary Table.

Sectors Status Register

**Figure 3-214. SECTSTAT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | RESERVED | | STATUS_SECTN | | STATUS_SECTM | |
| R-0h | | R-0h | | R-0h | | R-0h | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| STATUS_SECTL | | STATUS_SECTK | | STATUS_SECTJ | | STATUS_SECTI | |
| R-0h | | R-0h | | R-0h | | R-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| STATUS_SECTH | | STATUS_SECTG | | STATUS_SECTF | | STATUS_SECTE | |
| R-0h | | R-0h | | R-0h | | R-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| STATUS_SECTD | | STATUS_SECTC | | STATUS_SECTB | | STATUS_SECTA | |
| R-0h | | R-0h | | R-0h | | R-0h | |

**Table 3-239. SECTSTAT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | RESERVED | R | 0h | Reserved |
| 29-28 | RESERVED | R | 0h | Reserved |
| 27-26 | STATUS_SECTN | R | 0h | Reflects the status of flash sector N.<br>00 : Sector is in-accessible<br>01 : Sector belongs to Zone1.<br>10 : Sector belongs to Zone2.<br>11: Sector is un-secure and code running in both zone have full access to it.<br>Reset type: SYSRSn |
| 25-24 | STATUS_SECTM | R | 0h | Reflects the status of flash sector M.<br>00 : Sector is in-accessible<br>01 : Sector belongs to Zone1.<br>10 : Sector belongs to Zone2.<br>11: Sector is un-secure and code running in both zone have full access to it.<br>Reset type: SYSRSn |
| 23-22 | STATUS_SECTL | R | 0h | Reflects the status of flash sector L.<br>00 : Sector is in-accessible<br>01 : Sector belongs to Zone1.<br>10 : Sector belongs to Zone2.<br>11: Sector is un-secure and code running in both zone have full access to it.<br>Reset type: SYSRSn |
| 21-20 | STATUS_SECTK | R | 0h | Reflects the status of flash sector K.<br>00 : Sector is in-accessible<br>01 : Sector belongs to Zone1.<br>10 : Sector belongs to Zone2.<br>11: Sector is un-secure and code running in both zone have full access to it.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-239. SECTSTAT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19-18 | STATUS_SECTJ | R | 0h | Reflects the status of flash sector J.<br>00 : Sector is in-accessible<br>01 : Sector belongs to Zone1.<br>10 : Sector belongs to Zone2.<br>11: Sector is un-secure and code running in both zone have full access to it.<br>Reset type: SYSRSn |
| 17-16 | STATUS_SECTI | R | 0h | Reflects the status of flash sector I.<br>00 : Sector is in-accessible<br>01 : Sector belongs to Zone1.<br>10 : Sector belongs to Zone2.<br>11: Sector is un-secure and code running in both zone have full access to it.<br>Reset type: SYSRSn |
| 15-14 | STATUS_SECTH | R | 0h | Reflects the status of flash sector H.<br>00 : Sector is in-accessible<br>01 : Sector belongs to Zone1.<br>10 : Sector belongs to Zone2.<br>11: Sector is un-secure and code running in both zone have full access to it.<br>Reset type: SYSRSn |
| 13-12 | STATUS_SECTG | R | 0h | Reflects the status of flash sector G.<br>00 : Sector is in-accessible<br>01 : Sector belongs to Zone1.<br>10 : Sector belongs to Zone2.<br>11: Sector is un-secure and code running in both zone have full access to it.<br>Reset type: SYSRSn |
| 11-10 | STATUS_SECTF | R | 0h | Reflects the status of flash sector F.<br>00 : Sector is in-accessible<br>01 : Sector belongs to Zone1.<br>10 : Sector belongs to Zone2.<br>11: Sector is un-secure and code running in both zone have full access to it.<br>Reset type: SYSRSn |
| 9-8 | STATUS_SECTE | R | 0h | Reflects the status of flash sector E.<br>00 : Sector is in-accessible<br>01 : Sector belongs to Zone1.<br>10 : Sector belongs to Zone2.<br>11: Sector is un-secure and code running in both zone have full access to it.<br>Reset type: SYSRSn |
| 7-6 | STATUS_SECTD | R | 0h | Reflects the status of flash sector D.<br>00 : Sector is in-accessible<br>01 : Sector belongs to Zone1.<br>10 : Sector belongs to Zone2.<br>11: Sector is un-secure and code running in both zone have full access to it.<br>Reset type: SYSRSn |
| 5-4 | STATUS_SECTC | R | 0h | Reflects the status of flash sector C.<br>00 : Sector is in-accessible<br>01 : Sector belongs to Zone1.<br>10 : Sector belongs to Zone2.<br>11: Sector is un-secure and code running in both zone have full access to it.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-239. SECTSTAT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-2 | STATUS_SECTB | R | 0h | Reflects the status of flash sector B.<br>00 : Sector is in-accessible<br>01 : Sector belongs to Zone1.<br>10 : Sector belongs to Zone2.<br>11: Sector is un-secure and code running in both zone have full access to it.<br>Reset type: SYSRSn |
| 1-0 | STATUS_SECTA | R | 0h | Reflects the status of flash sector A.<br>00 : Sector is in-accessible<br>01 : Sector belongs to Zone1.<br>10 : Sector belongs to Zone2.<br>11: Sector is un-secure and code running in both zone have full access to it.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.16.3 RAMSTAT Register (Offset = 4h) [Reset = 00000000h]

RAMSTAT is shown in Figure 3-215 and described in Table 3-240.

Return to the Summary Table.

RAM Status Register

**Figure 3-215. RAMSTAT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | STATUS_CLA1 | | RESERVED | | | |
| R-0h | | R-0h | | R-0h | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| STATUS_RAM7 | | STATUS_RAM6 | | STATUS_RAM5 | | STATUS_RAM4 | |
| R-0h | | R-0h | | R-0h | | R-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| STATUS_RAM3 | | STATUS_RAM2 | | STATUS_RAM1 | | STATUS_RAM0 | |
| R-0h | | R-0h | | R-0h | | R-0h | |

**Table 3-240. RAMSTAT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | RESERVED | R | 0h | Reserved |
| 29-28 | STATUS_CLA1 | R | 0h | Reflects the status of CLA1.<br>00 : CLA is in-accessible<br>01 : CLA belongs to Zone1.<br>10 : CLA belongs to Zone2.<br>11: CLA is un-secure and code running in both zone have full access to it.<br>Reset type: SYSRSn |
| 27-16 | RESERVED | R | 0h | Reserved |
| 15-14 | STATUS_RAM7 | R | 0h | Reflects the status of D1 RAM.<br>00 : RAM is in-accessible<br>01 : RAM belongs to Zone1.<br>10 : RAM belongs to Zone2.<br>11: RAM is un-secure and code running in both zone have full access to it.<br>Reset type: SYSRSn |
| 13-12 | STATUS_RAM6 | R | 0h | Reflects the status of D0 RAM.<br>00 : RAM is in-accessible<br>01 : RAM belongs to Zone1.<br>10 : RAM belongs to Zone2.<br>11: RAM is un-secure and code running in both zone have full access to it.<br>Reset type: SYSRSn |
| 11-10 | STATUS_RAM5 | R | 0h | Reflects the status of LS5 RAM.<br>00 : RAM is in-accessible<br>01 : RAM belongs to Zone1.<br>10 : RAM belongs to Zone2.<br>11: RAM is un-secure and code running in both zone have full access to it.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-240. RAMSTAT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 9-8 | STATUS_RAM4 | R | 0h | Reflects the status of LS4 RAM.<br>00 : RAM is in-accessible<br>01 : RAM belongs to Zone1.<br>10 : RAM belongs to Zone2.<br>11: RAM is un-secure and code running in both zone have full access to it.<br>Reset type: SYSRSn |
| 7-6 | STATUS_RAM3 | R | 0h | Reflects the status of LS3 RAM.<br>00 : RAM is in-accessible<br>01 : RAM belongs to Zone1.<br>10 : RAM belongs to Zone2.<br>11: RAM is un-secure and code running in both zone have full access to it.<br>Reset type: SYSRSn |
| 5-4 | STATUS_RAM2 | R | 0h | Reflects the status of LS2 RAM.<br>00 : RAM is in-accessible<br>01 : RAM belongs to Zone1.<br>10 : RAM belongs to Zone2.<br>11: RAM is un-secure and code running in both zone have full access to it.<br>Reset type: SYSRSn |
| 3-2 | STATUS_RAM1 | R | 0h | Reflects the status of LS1 RAM.<br>00 : RAM is in-accessible<br>01 : RAM belongs to Zone1.<br>10 : RAM belongs to Zone2.<br>11: RAM is un-secure and code running in both zone have full access to it.<br>Reset type: SYSRSn |
| 1-0 | STATUS_RAM0 | R | 0h | Reflects the status of LS0 RAM.<br>00 : RAM is in-accessible<br>01 : RAM belongs to Zone1.<br>10 : RAM belongs to Zone2.<br>11: RAM is un-secure and code running in both zone have full access to it.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.17 MEM_CFG_REGS Registers

Table 3-241 lists the memory-mapped registers for the MEM_CFG_REGS registers. All register offset addresses not listed in Table 3-241 should be considered as reserved locations and the register contents should not be modified.

**Table 3-241. MEM_CFG_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | DxLOCK | Dedicated RAM Config Lock Register | EALLOW | Go |
| 2h | DxCOMMIT | Dedicated RAM Config Lock Commit Register | EALLOW | Go |
| 8h | DxACCPROT0 | Dedicated RAM Config Register | EALLOW | Go |
| 10h | DxTEST | Dedicated RAM TEST Register | EALLOW | Go |
| 12h | DxINIT | Dedicated RAM Init Register | EALLOW | Go |
| 14h | DxINITDONE | Dedicated RAM InitDone Status Register | | Go |
| 20h | LSxLOCK | Local Shared RAM Config Lock Register | EALLOW | Go |
| 22h | LSxCOMMIT | Local Shared RAM Config Lock Commit Register | EALLOW | Go |
| 24h | LSxMSEL | Local Shared RAM Master Sel Register | EALLOW | Go |
| 26h | LSxCLAPGM | Local Shared RAM Prog/Exe control Register | EALLOW | Go |
| 28h | LSxACCPROT0 | Local Shared RAM Config Register 0 | EALLOW | Go |
| 2Ah | LSxACCPROT1 | Local Shared RAM Config Register 1 | EALLOW | Go |
| 30h | LSxTEST | Local Shared RAM TEST Register | EALLOW | Go |
| 32h | LSxINIT | Local Shared RAM Init Register | EALLOW | Go |
| 34h | LSxINITDONE | Local Shared RAM InitDone Status Register | | Go |
| 40h | GSxLOCK | Global Shared RAM Config Lock Register | EALLOW | Go |
| 42h | GSxCOMMIT | Global Shared RAM Config Lock Commit Register | EALLOW | Go |
| 44h | GSxMSEL | Global Shared RAM Master Sel Register | EALLOW | Go |
| 48h | GSxACCPROT0 | Global Shared RAM Config Register 0 | EALLOW | Go |
| 4Ah | GSxACCPROT1 | Global Shared RAM Config Register 1 | EALLOW | Go |
| 4Ch | GSxACCPROT2 | Global Shared RAM Config Register 2 | EALLOW | Go |
| 4Eh | GSxACCPROT3 | Global Shared RAM Config Register 3 | EALLOW | Go |
| 50h | GSxTEST | Global Shared RAM TEST Register | EALLOW | Go |
| 52h | GSxINIT | Global Shared RAM Init Register | EALLOW | Go |
| 54h | GSxINITDONE | Global Shared RAM InitDone Status Register | | Go |
| 70h | MSGxTEST | Message RAM TEST Register | EALLOW | Go |
| 72h | MSGxINIT | Message RAM Init Register | EALLOW | Go |
| 74h | MSGxINITDONE | Message RAM InitDone Status Register | | Go |

Complex bit access types are encoded to fit into small table cells. Table 3-242 shows the codes that are used for access types in this section.

**Table 3-242. MEM_CFG_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| R-0 | R<br>-0 | Read<br>Returns 0s |
| Write Type | | |
| W | W | Write |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-242. MEM_CFG_REGS Access Type Codes (continued)**

| Access Type | Code | Description |
|---|---|---|
| W1S | W<br>1S | Write<br>1 to set |
| WSonce | W<br>Sonce | Write<br>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. |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.17.1 DxLOCK Register (Offset = 0h) [Reset = 00000000h]

DxLOCK is shown in Figure 3-216 and described in Table 3-243.

Return to the Summary Table.

Dedicated RAM Config Lock Register

**Figure 3-216. DxLOCK Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | LOCK_D1 | LOCK_D0 | RESERVED | |
| R-0h | | | | R/W-0h | R/W-0h | R-0h | |

**Table 3-243. DxLOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-4 | RESERVED | R | 0h | Reserved |
| 3 | LOCK_D1 | R/W | 0h | Locks the write to access protection and master select fields for D1 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed.<br>1: Write to ACCPROT and Mselect fields are blocked.<br>Reset type: SYSRSn |
| 2 | LOCK_D0 | R/W | 0h | Locks the write to access protection and master select fields for D0 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed.<br>1: Write to ACCPROT and Mselect fields are blocked.<br>Reset type: SYSRSn |
| 1-0 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.17.2 DxCOMMIT Register (Offset = 2h) [Reset = 00000000h]

DxCOMMIT is shown in Figure 3-217 and described in Table 3-244.

Return to the Summary Table.

Dedicated RAM Config Lock Commit Register

**Figure 3-217. DxCOMMIT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | COMMIT_D1 | COMMIT_D0 | RESERVED | |
| R-0h | | | | R/WSonce-0h | R/WSonce-0h | R-0h | |

**Table 3-244. DxCOMMIT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-4 | RESERVED | R | 0h | Reserved |
| 3 | COMMIT_D1 | R/WSonce | 0h | Permanently Locks the write to access protection and master select fields for D1 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in DxLOCK register.<br>1: Write to ACCPROT and Mselect fields are permanently blocked.<br>Reset type: SYSRSn |
| 2 | COMMIT_D0 | R/WSonce | 0h | Permanently Locks the write to access protection and master select fields for D0 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in DxLOCK register.<br>1: Write to ACCPROT and Mselect fields are permanently blocked.<br>Reset type: SYSRSn |
| 1-0 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.17.3 DxACCPROT0 Register (Offset = 8h) [Reset = 00000000h]

DxACCPROT0 is shown in Figure 3-218 and described in Table 3-245.

Return to the Summary Table.

Dedicated RAM Config Register

**Figure 3-218. DxACCPROT0 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | CPUWRPROT_D1 | FETCHPROT_D1 |
| R-0h | | | | | | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | CPUWRPROT_D0 | FETCHPROT_D0 |
| R-0h | | | | | | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |

**Table 3-245. DxACCPROT0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-26 | RESERVED | R | 0h | Reserved |
| 25 | CPUWRPROT_D1 | R/W | 0h | CPU WR Protection For D1 RAM:<br>0: CPU Writes are allowed.<br>1: CPU Writes are blocked.<br>Reset type: SYSRSn |
| 24 | FETCHPROT_D1 | R/W | 0h | Fetch Protection For D1 RAM:<br>0: CPU Fetch are allowed.<br>1: CPU Fetch are blocked.<br>Reset type: SYSRSn |
| 23-18 | RESERVED | R | 0h | Reserved |
| 17 | CPUWRPROT_D0 | R/W | 0h | CPU WR Protection For D0 RAM:<br>0: CPU Writes are allowed.<br>1: CPU Writes are block.<br>Reset type: SYSRSn |
| 16 | FETCHPROT_D0 | R/W | 0h | Fetch Protection For D0 RAM:<br>0: CPU Fetch are allowed.<br>1: CPU Fetch are blocked.<br>Reset type: SYSRSn |
| 15-0 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.17.4 DxTEST Register (Offset = 10h) [Reset = 00000000h]

DxTEST is shown in Figure 3-219 and described in Table 3-246.

Return to the Summary Table.

Dedicated RAM TEST Register

**Figure 3-219. DxTEST Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TEST_D1 | | TEST_D0 | | TEST_M1 | | TEST_M0 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 3-246. DxTEST Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-8 | RESERVED | R | 0h | Reserved |
| 7-6 | TEST_D1 | R/W | 0h | Selects the different modes for D1 RAM:<br>00: Functional Mode.<br>01: Writes are allowed to data bits only. No write to ECC bits.<br>10: Writes are allowed to ECC bits only. No write to data bits.<br>11: Functional Mode.<br>Reset type: SYSRSn |
| 5-4 | TEST_D0 | R/W | 0h | Selects the different modes for D0 RAM:<br>00: Functional Mode.<br>01: Writes are allowed to data bits only. No write to ECC bits.<br>10: Writes are allowed to ECC bits only. No write to data bits.<br>11: Functional Mode.<br>Reset type: SYSRSn |
| 3-2 | TEST_M1 | R/W | 0h | Selects the different modes for M1 RAM:<br>00: Functional Mode.<br>01: Writes are allowed to data bits only. No write to ECC bits.<br>10: Writes are allowed to ECC bits only. No write to data bits.<br>11: Functional Mode.<br>Reset type: SYSRSn |
| 1-0 | TEST_M0 | R/W | 0h | Selects the different modes for M0 RAM:<br>00: Functional Mode.<br>01: Writes are allowed to data bits only. No write to ECC bits.<br>10: Writes are allowed to ECC bits only. No write to data bits.<br>11: Functional Mode.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.17.5 DxINIT Register (Offset = 12h) [Reset = 00000000h]

DxINIT is shown in Figure 3-220 and described in Table 3-247.

Return to the Summary Table.

Dedicated RAM Init Register

**Figure 3-220. DxINIT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | INIT_D1 | INIT_D0 | INIT_M1 | INIT_M0 |
| R-0h | | | | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h |

**Table 3-247. DxINIT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-4 | RESERVED | R | 0h | Reserved |
| 3 | INIT_D1 | R-0/W1S | 0h | RAM Initialization control for D1 RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |
| 2 | INIT_D0 | R-0/W1S | 0h | RAM Initialization control for D0 RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |
| 1 | INIT_M1 | R-0/W1S | 0h | RAM Initialization control for M1 RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |
| 0 | INIT_M0 | R-0/W1S | 0h | RAM Initialization control for M0 RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.17.6 DxINITDONE Register (Offset = 14h) [Reset = 00000000h]

DxINITDONE is shown in Figure 3-221 and described in Table 3-248.

Return to the Summary Table.

Dedicated RAM InitDone Status Register

**Figure 3-221. DxINITDONE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | INITDONE_D1 | INITDONE_D0 | INITDONE_M1 | INITDONE_M0 |
| R-0h | | | | R-0h | R-0h | R-0h | R-0h |

**Table 3-248. DxINITDONE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-4 | RESERVED | R | 0h | Reserved |
| 3 | INITDONE_D1 | R | 0h | RAM Initialization status for D1 RAM:<br>0: RAM Initialization is not done.<br>1: RAM Initialization is done.<br>Reset type: SYSRSn |
| 2 | INITDONE_D0 | R | 0h | RAM Initialization status for D0 RAM:<br>0: RAM Initialization is not done.<br>1: RAM Initialization is done.<br>Reset type: SYSRSn |
| 1 | INITDONE_M1 | R | 0h | RAM Initialization status for M1 RAM:<br>0: RAM Initialization is not done.<br>1: RAM Initialization is done.<br>Reset type: SYSRSn |
| 0 | INITDONE_M0 | R | 0h | RAM Initialization status for M0 RAM:<br>0: RAM Initialization is not done.<br>1: RAM Initialization is done.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.17.7 LSxLOCK Register (Offset = 20h) [Reset = 00000000h]

LSxLOCK is shown in Figure 3-222 and described in Table 3-249.

Return to the Summary Table.

Local Shared RAM Config Lock Register

**Figure 3-222. LSxLOCK Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | LOCK_LS5 | LOCK_LS4 | LOCK_LS3 | LOCK_LS2 | LOCK_LS1 | LOCK_LS0 |
| R-0h | | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-249. LSxLOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-6 | RESERVED | R | 0h | Reserved |
| 5 | LOCK_LS5 | R/W | 0h | Locks the write to access protection and master select fields for LS5 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed.<br>1: Write to ACCPROT and Mselect fields are blocked.<br>Reset type: SYSRSn |
| 4 | LOCK_LS4 | R/W | 0h | Locks the write to access protection and master select fields for LS4 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed.<br>1: Write to ACCPROT and Mselect fields are blocked.<br>Reset type: SYSRSn |
| 3 | LOCK_LS3 | R/W | 0h | Locks the write to access protection and master select fields for LS3 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed.<br>1: Write to ACCPROT and Mselect fields are blocked.<br>Reset type: SYSRSn |
| 2 | LOCK_LS2 | R/W | 0h | Locks the write to access protection and master select fields for LS2 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed.<br>1: Write to ACCPROT and Mselect fields are blocked.<br>Reset type: SYSRSn |
| 1 | LOCK_LS1 | R/W | 0h | Locks the write to access protection and master select fields for LS1 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed.<br>1: Write to ACCPROT and Mselect fields are blocked.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-249. LSxLOCK Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | LOCK_LS0 | R/W | 0h | Locks the write to access protection and master select fields for LS0 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed.<br>1: Write to ACCPROT and Mselect fields are blocked.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.17.8 LSxCOMMIT Register (Offset = 22h) [Reset = 00000000h]

LSxCOMMIT is shown in Figure 3-223 and described in Table 3-250.

Return to the Summary Table.

Local Shared RAM Config Lock Commit Register

**Figure 3-223. LSxCOMMIT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | COMMIT_LS5 | COMMIT_LS4 | COMMIT_LS3 | COMMIT_LS2 | COMMIT_LS1 | COMMIT_LS0 |
| R-0h | | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h |

**Table 3-250. LSxCOMMIT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-6 | RESERVED | R | 0h | Reserved |
| 5 | COMMIT_LS5 | R/WSonce | 0h | Permanently Locks the write to access protection and master select fields for LS5 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in LSxLOCK register.<br>1: Write to ACCPROT and Mselect fields are permanently blocked.<br>Reset type: SYSRSn |
| 4 | COMMIT_LS4 | R/WSonce | 0h | Permanently Locks the write to access protection and master select fields for LS4 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in LSxLOCK register.<br>1: Write to ACCPROT and Mselect fields are permanently blocked.<br>Reset type: SYSRSn |
| 3 | COMMIT_LS3 | R/WSonce | 0h | Permanently Locks the write to access protection and master select fields for LS3 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in LSxLOCK register.<br>1: Write to ACCPROT and Mselect fields are permanently blocked.<br>Reset type: SYSRSn |
| 2 | COMMIT_LS2 | R/WSonce | 0h | Permanently Locks the write to access protection and master select fields for LS2 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in LSxLOCK register.<br>1: Write to ACCPROT and Mselect fields are permanently blocked.<br>Reset type: SYSRSn |
| 1 | COMMIT_LS1 | R/WSonce | 0h | Permanently Locks the write to access protection and master select fields for LS1 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in LSxLOCK register.<br>1: Write to ACCPROT and Mselect fields are permanently blocked.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-250. LSxCOMMIT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | COMMIT_LS0 | R/WSonce | 0h | Permanently Locks the write to access protection and master select fields for LS0 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in LSxLOCK register.<br>1: Write to ACCPROT and Mselect fields are permanently blocked.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.17.9 LSxMSEL Register (Offset = 24h) [Reset = 00000000h]

LSxMSEL is shown in Figure 3-224 and described in Table 3-251.

Return to the Summary Table.

Local Shared RAM Master Sel Register

**Figure 3-224. LSxMSEL Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | MSEL_LS5 | | MSEL_LS4 | |
| R-0h | | | | R/W-0h | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MSEL_LS3 | | MSEL_LS2 | | MSEL_LS1 | | MSEL_LS0 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 3-251. LSxMSEL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-12 | RESERVED | R | 0h | Reserved |
| 11-10 | MSEL_LS5 | R/W | 0h | Master Select for LS5 RAM:<br>00: Memory is dedicated to CPU.<br>01: Memory is shared between CPU and CLA1.<br>10: Reserved.<br>11: Reserved.<br>Reset type: SYSRSn |
| 9-8 | MSEL_LS4 | R/W | 0h | Master Select for LS4 RAM:<br>00: Memory is dedicated to CPU.<br>01: Memory is shared between CPU and CLA1.<br>10: Reserved.<br>11: Reserved.<br>Reset type: SYSRSn |
| 7-6 | MSEL_LS3 | R/W | 0h | Master Select for LS3 RAM:<br>00: Memory is dedicated to CPU.<br>01: Memory is shared between CPU and CLA1.<br>10: Reserved.<br>11: Reserved.<br>Reset type: SYSRSn |
| 5-4 | MSEL_LS2 | R/W | 0h | Master Select for LS2 RAM:<br>00: Memory is dedicated to CPU.<br>01: Memory is shared between CPU and CLA1.<br>10: Reserved.<br>11: Reserved.<br>Reset type: SYSRSn |
| 3-2 | MSEL_LS1 | R/W | 0h | Master Select for LS1 RAM:<br>00: Memory is dedicated to CPU.<br>01: Memory is shared between CPU and CLA1.<br>10: Reserved.<br>11: Reserved.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-251. LSxMSEL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1-0 | MSEL_LS0 | R/W | 0h | Master Select for LS0 RAM:<br>00: Memory is dedicated to CPU.<br>01: Memory is shared between CPU and CLA1.<br>10: Reserved.<br>11: Reserved.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.17.10 LSxCLAPGM Register (Offset = 26h) [Reset = 00000000h]

LSxCLAPGM is shown in Figure 3-225 and described in Table 3-252.

Return to the Summary Table.

Local Shared RAM Prog/Exe control Register

**Figure 3-225. LSxCLAPGM Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | CLAPGM_LS5 | CLAPGM_LS4 | CLAPGM_LS3 | CLAPGM_LS2 | CLAPGM_LS1 | CLAPGM_LS0 |
| R-0h | | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-252. LSxCLAPGM Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-6 | RESERVED | R | 0h | Reserved |
| 5 | CLAPGM_LS5 | R/W | 0h | Selects LS5 RAM as program vs data memory for CLA:<br>0: CLA Data memory.<br>1: CLA Program memory.<br>Reset type: SYSRSn |
| 4 | CLAPGM_LS4 | R/W | 0h | Selects LS4 RAM as program vs data memory for CLA:<br>0: CLA Data memory.<br>1: CLA Program memory.<br>Reset type: SYSRSn |
| 3 | CLAPGM_LS3 | R/W | 0h | Selects LS3 RAM as program vs data memory for CLA:<br>0: CLA Data memory.<br>1: CLA Program memory.<br>Reset type: SYSRSn |
| 2 | CLAPGM_LS2 | R/W | 0h | Selects LS2 RAM as program vs data memory for CLA:<br>0: CLA Data memory.<br>1: CLA Program memory.<br>Reset type: SYSRSn |
| 1 | CLAPGM_LS1 | R/W | 0h | Selects LS1 RAM as program vs data memory for CLA:<br>0: CLA Data memory.<br>1: CLA Program memory.<br>Reset type: SYSRSn |
| 0 | CLAPGM_LS0 | R/W | 0h | Selects LS0 RAM as program vs data memory for CLA:<br>0: CLA Data memory.<br>1: CLA Program memory.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.17.11 LSxACCPROT0 Register (Offset = 28h) [Reset = 00000000h]

LSxACCPROT0 is shown in Figure 3-226 and described in Table 3-253.

Return to the Summary Table.

Local Shared RAM Config Register 0

**Figure 3-226. LSxACCPROT0 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | CPUWRPROT_LS3 | FETCHPROT_LS3 |
| R-0h | | | | | | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | CPUWRPROT_LS2 | FETCHPROT_LS2 |
| R-0h | | | | | | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | CPUWRPROT_LS1 | FETCHPROT_LS1 |
| R-0h | | | | | | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | CPUWRPROT_LS0 | FETCHPROT_LS0 |
| R-0h | | | | | | R/W-0h | R/W-0h |

**Table 3-253. LSxACCPROT0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-26 | RESERVED | R | 0h | Reserved |
| 25 | CPUWRPROT_LS3 | R/W | 0h | CPU WR Protection For LS3 RAM:<br>0: CPU Writes are allowed.<br>1: CPU Writes are blocked.<br>Reset type: SYSRSn |
| 24 | FETCHPROT_LS3 | R/W | 0h | Fetch Protection For LS3 RAM:<br>0: CPU Fetch are allowed.<br>1: CPU Fetch are blocked.<br>Reset type: SYSRSn |
| 23-18 | RESERVED | R | 0h | Reserved |
| 17 | CPUWRPROT_LS2 | R/W | 0h | CPU WR Protection For LS2 RAM:<br>0: CPU Writes are allowed.<br>1: CPU Writes are blocked.<br>Reset type: SYSRSn |
| 16 | FETCHPROT_LS2 | R/W | 0h | Fetch Protection For LS2 RAM:<br>0: CPU Fetch are allowed.<br>1: CPU Fetch are blocked.<br>Reset type: SYSRSn |
| 15-10 | RESERVED | R | 0h | Reserved |
| 9 | CPUWRPROT_LS1 | R/W | 0h | CPU WR Protection For LS1 RAM:<br>0: CPU Writes are allowed.<br>1: CPU Writes are blocked.<br>Reset type: SYSRSn |
| 8 | FETCHPROT_LS1 | R/W | 0h | Fetch Protection For LS1 RAM:<br>0: CPU Fetch are allowed.<br>1: CPU Fetch are blocked.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-253. LSxACCPROT0 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 7-2 | RESERVED | R | 0h | Reserved |
| 1 | CPUWRPROT_LS0 | R/W | 0h | CPU WR Protection For LS0 RAM:<br>0: CPU Writes are allowed.<br>1: CPU Writes are blocked.<br>Reset type: SYSRSn |
| 0 | FETCHPROT_LS0 | R/W | 0h | Fetch Protection For LS0 RAM:<br>0: CPU Fetch are allowed.<br>1: CPU Fetch are blocked.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.17.12 LSxACCPROT1 Register (Offset = 2Ah) [Reset = 00000000h]

LSxACCPROT1 is shown in Figure 3-227 and described in Table 3-254.

Return to the Summary Table.

Local Shared RAM Config Register 1

**Figure 3-227. LSxACCPROT1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | CPUWRPROT_LS5 | FETCHPROT_LS5 |
| R-0h | | | | | | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | CPUWRPROT_LS4 | FETCHPROT_LS4 |
| R-0h | | | | | | R/W-0h | R/W-0h |

**Table 3-254. LSxACCPROT1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-10 | RESERVED | R | 0h | Reserved |
| 9 | CPUWRPROT_LS5 | R/W | 0h | CPU WR Protection For LS5 RAM:<br>0: CPU Writes are allowed.<br>1: CPU Writes are blocked.<br>Reset type: SYSRSn |
| 8 | FETCHPROT_LS5 | R/W | 0h | Fetch Protection For LS5 RAM:<br>0: CPU Fetch are allowed.<br>1: CPU Fetch are blocked.<br>Reset type: SYSRSn |
| 7-2 | RESERVED | R | 0h | Reserved |
| 1 | CPUWRPROT_LS4 | R/W | 0h | CPU WR Protection For LS4 RAM:<br>0: CPU Writes are allowed.<br>1: CPU Writes are blocked.<br>Reset type: SYSRSn |
| 0 | FETCHPROT_LS4 | R/W | 0h | Fetch Protection For LS4 RAM:<br>0: CPU Fetch are allowed.<br>1: CPU Fetch are blocked.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.17.13 LSxTEST Register (Offset = 30h) [Reset = 00000000h]

LSxTEST is shown in Figure 3-228 and described in Table 3-255.

Return to the Summary Table.

Local Shared RAM TEST Register

**Figure 3-228. LSxTEST Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | TEST_LS5 | | TEST_LS4 | |
| R-0h | | | | R/W-0h | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TEST_LS3 | | TEST_LS2 | | TEST_LS1 | | TEST_LS0 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 3-255. LSxTEST Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-12 | RESERVED | R | 0h | Reserved |
| 11-10 | TEST_LS5 | R/W | 0h | Selects the different modes for LS5 RAM:<br>00: Functional Mode.<br>01: Writes are allowed to data bits only. No write to parity bits.<br>10: Writes are allowed to parity bits only. No write to data bits.<br>11: Functional Mode.<br>Reset type: SYSRSn |
| 9-8 | TEST_LS4 | R/W | 0h | Selects the different modes for LS4 RAM:<br>00: Functional Mode.<br>01: Writes are allowed to data bits only. No write to parity bits.<br>10: Writes are allowed to parity bits only. No write to data bits.<br>11: Functional Mode.<br>Reset type: SYSRSn |
| 7-6 | TEST_LS3 | R/W | 0h | Selects the different modes for LS3 RAM:<br>00: Functional Mode.<br>01: Writes are allowed to data bits only. No write to parity bits.<br>10: Writes are allowed to parity bits only. No write to data bits.<br>11: Functional Mode.<br>Reset type: SYSRSn |
| 5-4 | TEST_LS2 | R/W | 0h | Selects the different modes for LS2 RAM:<br>00: Functional Mode.<br>01: Writes are allowed to data bits only. No write to parity bits.<br>10: Writes are allowed to parity bits only. No write to data bits.<br>11: Functional Mode.<br>Reset type: SYSRSn |
| 3-2 | TEST_LS1 | R/W | 0h | Selects the different modes for LS1 RAM:<br>00: Functional Mode.<br>01: Writes are allowed to data bits only. No write to parity bits.<br>10: Writes are allowed to parity bits only. No write to data bits.<br>11: Functional Mode.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-255. LSxTEST Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1-0 | TEST_LS0 | R/W | 0h | Selects the different modes for LS0 RAM:<br>00: Functional Mode.<br>01: Writes are allowed to data bits only. No write to parity bits.<br>10: Writes are allowed to parity bits only. No write to data bits.<br>11: Functional Mode.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.17.14 LSxINIT Register (Offset = 32h) [Reset = 00000000h]

LSxINIT is shown in Figure 3-229 and described in Table 3-256.

Return to the Summary Table.

Local Shared RAM Init Register

**Figure 3-229. LSxINIT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | INIT_LS5 | INIT_LS4 | INIT_LS3 | INIT_LS2 | INIT_LS1 | INIT_LS0 |
| R-0h | | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h |

**Table 3-256. LSxINIT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-6 | RESERVED | R | 0h | Reserved |
| 5 | INIT_LS5 | R-0/W1S | 0h | RAM Initialization control for LS5 RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |
| 4 | INIT_LS4 | R-0/W1S | 0h | RAM Initialization control for LS4 RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |
| 3 | INIT_LS3 | R-0/W1S | 0h | RAM Initialization control for LS3 RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |
| 2 | INIT_LS2 | R-0/W1S | 0h | RAM Initialization control for LS2 RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |
| 1 | INIT_LS1 | R-0/W1S | 0h | RAM Initialization control for LS1 RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |
| 0 | INIT_LS0 | R-0/W1S | 0h | RAM Initialization control for LS0 RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.17.15 LSxINITDONE Register (Offset = 34h) [Reset = 00000000h]

LSxINITDONE is shown in Figure 3-230 and described in Table 3-257.

Return to the Summary Table.

Local Shared RAM InitDone Status Register

**Figure 3-230. LSxINITDONE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | INITDONE_LS5 | INITDONE_LS4 | INITDONE_LS3 | INITDONE_LS2 | INITDONE_LS1 | INITDONE_LS0 |
| R-0h | | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 3-257. LSxINITDONE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-6 | RESERVED | R | 0h | Reserved |
| 5 | INITDONE_LS5 | R | 0h | RAM Initialization status for LS5 RAM:<br>0: RAM Initialization is not done.<br>1: RAM Initialization is done.<br>Reset type: SYSRSn |
| 4 | INITDONE_LS4 | R | 0h | RAM Initialization status for LS4 RAM:<br>0: RAM Initialization is not done.<br>1: RAM Initialization is done.<br>Reset type: SYSRSn |
| 3 | INITDONE_LS3 | R | 0h | RAM Initialization status for LS3 RAM:<br>0: RAM Initialization is not done.<br>1: RAM Initialization is done.<br>Reset type: SYSRSn |
| 2 | INITDONE_LS2 | R | 0h | RAM Initialization status for LS2 RAM:<br>0: RAM Initialization is not done.<br>1: RAM Initialization is done.<br>Reset type: SYSRSn |
| 1 | INITDONE_LS1 | R | 0h | RAM Initialization status for LS1 RAM:<br>0: RAM Initialization is not done.<br>1: RAM Initialization is done.<br>Reset type: SYSRSn |
| 0 | INITDONE_LS0 | R | 0h | RAM Initialization status for LS0 RAM:<br>0: RAM Initialization is not done.<br>1: RAM Initialization is done.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.17.16 GSxLOCK Register (Offset = 40h) [Reset = 00000000h]

GSxLOCK is shown in Figure 3-231 and described in Table 3-258.

Return to the Summary Table.

Global Shared RAM Config Lock Register

**Figure 3-231. GSxLOCK Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| LOCK_GS15 | LOCK_GS14 | LOCK_GS13 | LOCK_GS12 | LOCK_GS11 | LOCK_GS10 | LOCK_GS9 | LOCK_GS8 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| LOCK_GS7 | LOCK_GS6 | LOCK_GS5 | LOCK_GS4 | LOCK_GS3 | LOCK_GS2 | LOCK_GS1 | LOCK_GS0 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-258. GSxLOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15 | LOCK_GS15 | R/W | 0h | Locks the write to access protection and master select fields for GS15 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed.<br>1: Write to ACCPROT and Mselect fields are blocked.<br>Reset type: SYSRSn |
| 14 | LOCK_GS14 | R/W | 0h | Locks the write to access protection and master select fields for GS14 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed.<br>1: Write to ACCPROT and Mselect fields are blocked.<br>Reset type: SYSRSn |
| 13 | LOCK_GS13 | R/W | 0h | Locks the write to access protection and master select fields for GS13 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed.<br>1: Write to ACCPROT and Mselect fields are blocked.<br>Reset type: SYSRSn |
| 12 | LOCK_GS12 | R/W | 0h | Locks the write to access protection and master select fields for GS12 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed.<br>1: Write to ACCPROT and Mselect fields are blocked.<br>Reset type: SYSRSn |
| 11 | LOCK_GS11 | R/W | 0h | Locks the write to access protection and master select fields for GS11 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed.<br>1: Write to ACCPROT and Mselect fields are blocked.<br>Reset type: SYSRSn |
| 10 | LOCK_GS10 | R/W | 0h | Locks the write to access protection and master select fields for GS10 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed.<br>1: Write to ACCPROT and Mselect fields are blocked.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-258. GSxLOCK Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 9 | LOCK_GS9 | R/W | 0h | Locks the write to access protection and master select fields for GS9 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed.<br>1: Write to ACCPROT and Mselect fields are blocked.<br>Reset type: SYSRSn |
| 8 | LOCK_GS8 | R/W | 0h | Locks the write to access protection and master select fields for GS8 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed.<br>1: Write to ACCPROT and Mselect fields are blocked.<br>Reset type: SYSRSn |
| 7 | LOCK_GS7 | R/W | 0h | Locks the write to access protection and master select fields for GS7 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed.<br>1: Write to ACCPROT and Mselect fields are blocked.<br>Reset type: SYSRSn |
| 6 | LOCK_GS6 | R/W | 0h | Locks the write to access protection and master select fields for GS6 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed.<br>1: Write to ACCPROT and Mselect fields are blocked.<br>Reset type: SYSRSn |
| 5 | LOCK_GS5 | R/W | 0h | Locks the write to access protection and master select fields for GS5 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed.<br>1: Write to ACCPROT and Mselect fields are blocked.<br>Reset type: SYSRSn |
| 4 | LOCK_GS4 | R/W | 0h | Locks the write to access protection and master select fields for GS4 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed.<br>1: Write to ACCPROT and Mselect fields are blocked.<br>Reset type: SYSRSn |
| 3 | LOCK_GS3 | R/W | 0h | Locks the write to access protection and master select fields for GS3 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed.<br>1: Write to ACCPROT and Mselect fields are blocked.<br>Reset type: SYSRSn |
| 2 | LOCK_GS2 | R/W | 0h | Locks the write to access protection and master select fields for GS2 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed.<br>1: Write to ACCPROT and Mselect fields are blocked.<br>Reset type: SYSRSn |
| 1 | LOCK_GS1 | R/W | 0h | Locks the write to access protection and master select fields for GS1 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed.<br>1: Write to ACCPROT and Mselect fields are blocked.<br>Reset type: SYSRSn |
| 0 | LOCK_GS0 | R/W | 0h | Locks the write to access protection and master select fields for GS0 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed.<br>1: Write to ACCPROT and Mselect fields are blocked.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.17.17 GSxCOMMIT Register (Offset = 42h) [Reset = 00000000h]

GSxCOMMIT is shown in Figure 3-232 and described in Table 3-259.

Return to the Summary Table.

Global Shared RAM Config Lock Commit Register

**Figure 3-232. GSxCOMMIT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| COMMIT_GS15 | COMMIT_GS14 | COMMIT_GS13 | COMMIT_GS12 | COMMIT_GS11 | COMMIT_GS10 | COMMIT_GS9 | COMMIT_GS8 |
| R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| COMMIT_GS7 | COMMIT_GS6 | COMMIT_GS5 | COMMIT_GS4 | COMMIT_GS3 | COMMIT_GS2 | COMMIT_GS1 | COMMIT_GS0 |
| R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h |

**Table 3-259. GSxCOMMIT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15 | COMMIT_GS15 | R/WSonce | 0h | Permanently Locks the write to access protection and master select fields for GS15 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in GSxLOCK register.<br>1: Write to ACCPROT and Mselect fields are permanently blocked.<br>Reset type: SYSRSn |
| 14 | COMMIT_GS14 | R/WSonce | 0h | Permanently Locks the write to access protection and master select fields for GS14 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in GSxLOCK register.<br>1: Write to ACCPROT and Mselect fields are permanently blocked.<br>Reset type: SYSRSn |
| 13 | COMMIT_GS13 | R/WSonce | 0h | Permanently Locks the write to access protection and master select fields for GS13 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in GSxLOCK register.<br>1: Write to ACCPROT and Mselect fields are permanently blocked.<br>Reset type: SYSRSn |
| 12 | COMMIT_GS12 | R/WSonce | 0h | Permanently Locks the write to access protection and master select fields for GS12 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in GSxLOCK register.<br>1: Write to ACCPROT and Mselect fields are permanently blocked.<br>Reset type: SYSRSn |
| 11 | COMMIT_GS11 | R/WSonce | 0h | Permanently Locks the write to access protection and master select fields for GS11 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in GSxLOCK register.<br>1: Write to ACCPROT and Mselect fields are permanently blocked.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-259. GSxCOMMIT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 10 | COMMIT_GS10 | R/WSonce | 0h | Permanently Locks the write to access protection and master select fields for GS10 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in GSxLOCK register.<br>1: Write to ACCPROT and Mselect fields are permanently blocked.<br>Reset type: SYSRSn |
| 9 | COMMIT_GS9 | R/WSonce | 0h | Permanently Locks the write to access protection and master select fields for GS9 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in GSxLOCK register.<br>1: Write to ACCPROT and Mselect fields are permanently blocked.<br>Reset type: SYSRSn |
| 8 | COMMIT_GS8 | R/WSonce | 0h | Permanently Locks the write to access protection and master select fields for GS2 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in GSxLOCK register.<br>1: Write to ACCPROT and Mselect fields are permanently blocked.<br>Reset type: SYSRSn |
| 7 | COMMIT_GS7 | R/WSonce | 0h | Permanently Locks the write to access protection and master select fields for GS7 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in GSxLOCK register.<br>1: Write to ACCPROT and Mselect fields are permanently blocked.<br>Reset type: SYSRSn |
| 6 | COMMIT_GS6 | R/WSonce | 0h | Permanently Locks the write to access protection and master select fields for GS6 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in GSxLOCK register.<br>1: Write to ACCPROT and Mselect fields are permanently blocked.<br>Reset type: SYSRSn |
| 5 | COMMIT_GS5 | R/WSonce | 0h | Permanently Locks the write to access protection and master select fields for GS5 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in GSxLOCK register.<br>1: Write to ACCPROT and Mselect fields are permanently blocked.<br>Reset type: SYSRSn |
| 4 | COMMIT_GS4 | R/WSonce | 0h | Permanently Locks the write to access protection and master select fields for GS4 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in GSxLOCK register.<br>1: Write to ACCPROT and Mselect fields are permanently blocked.<br>Reset type: SYSRSn |
| 3 | COMMIT_GS3 | R/WSonce | 0h | Permanently Locks the write to access protection and master select fields for GS3 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in GSxLOCK register.<br>1: Write to ACCPROT and Mselect fields are permanently blocked.<br>Reset type: SYSRSn |
| 2 | COMMIT_GS2 | R/WSonce | 0h | Permanently Locks the write to access protection and master select fields for GS2 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in GSxLOCK register.<br>1: Write to ACCPROT and Mselect fields are permanently blocked.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-259. GSxCOMMIT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1 | COMMIT_GS1 | R/WSonce | 0h | Permanently Locks the write to access protection and master select fields for GS1 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in GSxLOCK register.<br>1: Write to ACCPROT and Mselect fields are permanently blocked.<br>Reset type: SYSRSn |
| 0 | COMMIT_GS0 | R/WSonce | 0h | Permanently Locks the write to access protection and master select fields for GS0 RAM:<br>0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in GSxLOCK register.<br>1: Write to ACCPROT and Mselect fields are permanently blocked.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.17.18 GSxMSEL Register (Offset = 44h) [Reset = 00000000h]

GSxMSEL is shown in Figure 3-233 and described in Table 3-260.

Return to the Summary Table.

Global Shared RAM Master Sel Register

**Figure 3-233. GSxMSEL Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MSEL_GS15 | MSEL_GS14 | MSEL_GS13 | MSEL_GS12 | MSEL_GS11 | MSEL_GS10 | MSEL_GS9 | MSEL_GS8 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MSEL_GS7 | MSEL_GS6 | MSEL_GS5 | MSEL_GS4 | MSEL_GS3 | MSEL_GS2 | MSEL_GS1 | MSEL_GS0 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-260. GSxMSEL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15 | MSEL_GS15 | R/W | 0h | Master Select for GS15 RAM:<br>0: CPU1 is master for this memory.<br>1: CPU2 is master for this memory.<br>Reset type: CPU1.SYSRSn |
| 14 | MSEL_GS14 | R/W | 0h | Master Select for GS14 RAM:<br>0: CPU1 is master for this memory.<br>1: CPU2 is master for this memory.<br>Reset type: CPU1.SYSRSn |
| 13 | MSEL_GS13 | R/W | 0h | Master Select for GS13 RAM:<br>0: CPU1 is master for this memory.<br>1: CPU2 is master for this memory.<br>Reset type: CPU1.SYSRSn |
| 12 | MSEL_GS12 | R/W | 0h | Master Select for GS12 RAM:<br>0: CPU1 is master for this memory.<br>1: CPU2 is master for this memory.<br>Reset type: CPU1.SYSRSn |
| 11 | MSEL_GS11 | R/W | 0h | Master Select for GS11 RAM:<br>0: CPU1 is master for this memory.<br>1: CPU2 is master for this memory.<br>Reset type: CPU1.SYSRSn |
| 10 | MSEL_GS10 | R/W | 0h | Master Select for GS10 RAM:<br>0: CPU1 is master for this memory.<br>1: CPU2 is master for this memory.<br>Reset type: CPU1.SYSRSn |
| 9 | MSEL_GS9 | R/W | 0h | Master Select for GS9 RAM:<br>0: CPU1 is master for this memory.<br>1: CPU2 is master for this memory.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-260. GSxMSEL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 8 | MSEL_GS8 | R/W | 0h | Master Select for GS8 RAM:<br>0: CPU1 is master for this memory.<br>1: CPU2 is master for this memory.<br>Reset type: CPU1.SYSRSn |
| 7 | MSEL_GS7 | R/W | 0h | Master Select for GS7 RAM:<br>0: CPU1 is master for this memory.<br>1: CPU2 is master for this memory.<br>Reset type: CPU1.SYSRSn |
| 6 | MSEL_GS6 | R/W | 0h | Master Select for GS6 RAM:<br>0: CPU1 is master for this memory.<br>1: CPU2 is master for this memory.<br>Reset type: CPU1.SYSRSn |
| 5 | MSEL_GS5 | R/W | 0h | Master Select for GS5 RAM:<br>0: CPU1 is master for this memory.<br>1: CPU2 is master for this memory.<br>Reset type: CPU1.SYSRSn |
| 4 | MSEL_GS4 | R/W | 0h | Master Select for GS4 RAM:<br>0: CPU1 is master for this memory.<br>1: CPU2 is master for this memory.<br>Reset type: CPU1.SYSRSn |
| 3 | MSEL_GS3 | R/W | 0h | Master Select for GS3 RAM:<br>0: CPU1 is master for this memory.<br>1: CPU2 is master for this memory.<br>Reset type: CPU1.SYSRSn |
| 2 | MSEL_GS2 | R/W | 0h | Master Select for GS2 RAM:<br>0: CPU1 is master for this memory.<br>1: CPU2 is master for this memory.<br>Reset type: CPU1.SYSRSn |
| 1 | MSEL_GS1 | R/W | 0h | Master Select for GS1 RAM:<br>0: CPU1 is master for this memory.<br>1: CPU2 is master for this memory.<br>Reset type: CPU1.SYSRSn |
| 0 | MSEL_GS0 | R/W | 0h | Master Select for GS0 RAM:<br>0: CPU1 is master for this memory.<br>1: CPU2 is master for this memory.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.17.19 GSxACCPROT0 Register (Offset = 48h) [Reset = 00000000h]

GSxACCPROT0 is shown in Figure 3-234 and described in Table 3-261.

Return to the Summary Table.

Global Shared RAM Config Register 0

**Figure 3-234. GSxACCPROT0 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | DMAWRPROT_GS3 | CPUWRPROT_GS3 | FETCHPROT_GS3 |
| R-0h | | | | | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | DMAWRPROT_GS2 | CPUWRPROT_GS2 | FETCHPROT_GS2 |
| R-0h | | | | | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | DMAWRPROT_GS1 | CPUWRPROT_GS1 | FETCHPROT_GS1 |
| R-0h | | | | | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | DMAWRPROT_GS0 | CPUWRPROT_GS0 | FETCHPROT_GS0 |
| R-0h | | | | | R/W-0h | R/W-0h | R/W-0h |

**Table 3-261. GSxACCPROT0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-27 | RESERVED | R | 0h | Reserved |
| 26 | DMAWRPROT_GS3 | R/W | 0h | DMA WR Protection For GS3 RAM:<br>0: DMA Writes are allowed.<br>1: DMA Writes are blocked.<br>Reset type: SYSRSn |
| 25 | CPUWRPROT_GS3 | R/W | 0h | CPU WR Protection For GS3 RAM:<br>0: CPU Writes are allowed.<br>1: CPU Writes are blocked.<br>Reset type: SYSRSn |
| 24 | FETCHPROT_GS3 | R/W | 0h | Fetch Protection For GS3 RAM:<br>0: CPU Fetch are allowed.<br>1: CPU Fetch are blocked.<br>Reset type: SYSRSn |
| 23-19 | RESERVED | R | 0h | Reserved |
| 18 | DMAWRPROT_GS2 | R/W | 0h | DMA WR Protection For GS2 RAM:<br>0: DMA Writes are allowed.<br>1: DMA Writes are blocked.<br>Reset type: SYSRSn |
| 17 | CPUWRPROT_GS2 | R/W | 0h | CPU WR Protection For GS2 RAM:<br>0: CPU Writes are allowed.<br>1: CPU Writes are blocked.<br>Reset type: SYSRSn |
| 16 | FETCHPROT_GS2 | R/W | 0h | Fetch Protection For GS2 RAM:<br>0: CPU Fetch are allowed.<br>1: CPU Fetch are blocked.<br>Reset type: SYSRSn |
| 15-11 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-261. GSxACCPROT0 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 10 | DMAWRPROT_GS1 | R/W | 0h | DMA WR Protection For GS1 RAM:<br>0: DMA Writes are allowed.<br>1: DMA Writes are blocked.<br>Reset type: SYSRSn |
| 9 | CPUWRPROT_GS1 | R/W | 0h | CPU WR Protection For GS1 RAM:<br>0: CPU Writes are allowed.<br>1: CPU Writes are blocked.<br>Reset type: SYSRSn |
| 8 | FETCHPROT_GS1 | R/W | 0h | Fetch Protection For GS1 RAM:<br>0: CPU Fetch are allowed.<br>1: CPU Fetch are blocked.<br>Reset type: SYSRSn |
| 7-3 | RESERVED | R | 0h | Reserved |
| 2 | DMAWRPROT_GS0 | R/W | 0h | DMA WR Protection For GS0 RAM:<br>0: DMA Writes are allowed.<br>1: DMA Writes are blocked.<br>Reset type: SYSRSn |
| 1 | CPUWRPROT_GS0 | R/W | 0h | CPU WR Protection For GS0 RAM:<br>0: CPU Writes are allowed.<br>1: CPU Writes are blocked.<br>Reset type: SYSRSn |
| 0 | FETCHPROT_GS0 | R/W | 0h | Fetch Protection For GS0 RAM:<br>0: CPU Fetch are allowed.<br>1: CPU Fetch are blocked.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.17.20 GSxACCPROT1 Register (Offset = 4Ah) [Reset = 00000000h]

GSxACCPROT1 is shown in Figure 3-235 and described in Table 3-262.

Return to the Summary Table.

Global Shared RAM Config Register 1

**Figure 3-235. GSxACCPROT1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | DMAWRPROT_GS7 | CPUWRPROT_GS7 | FETCHPROT_GS7 |
| R-0h | | | | | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | DMAWRPROT_GS6 | CPUWRPROT_GS6 | FETCHPROT_GS6 |
| R-0h | | | | | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | DMAWRPROT_GS5 | CPUWRPROT_GS5 | FETCHPROT_GS5 |
| R-0h | | | | | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | DMAWRPROT_GS4 | CPUWRPROT_GS4 | FETCHPROT_GS4 |
| R-0h | | | | | R/W-0h | R/W-0h | R/W-0h |

**Table 3-262. GSxACCPROT1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-27 | RESERVED | R | 0h | Reserved |
| 26 | DMAWRPROT_GS7 | R/W | 0h | DMA WR Protection For GS7 RAM:<br>0: DMA Writes are allowed.<br>1: DMA Writes are blocked.<br>Reset type: SYSRSn |
| 25 | CPUWRPROT_GS7 | R/W | 0h | CPU WR Protection For GS7 RAM:<br>0: CPU Writes are allowed.<br>1: CPU Writes are blocked.<br>Reset type: SYSRSn |
| 24 | FETCHPROT_GS7 | R/W | 0h | Fetch Protection For GS7 RAM:<br>0: CPU Fetch are allowed.<br>1: CPU Fetch are blocked.<br>Reset type: SYSRSn |
| 23-19 | RESERVED | R | 0h | Reserved |
| 18 | DMAWRPROT_GS6 | R/W | 0h | DMA WR Protection For GS6 RAM:<br>0: DMA Writes are allowed.<br>1: DMA Writes are blocked.<br>Reset type: SYSRSn |
| 17 | CPUWRPROT_GS6 | R/W | 0h | CPU WR Protection For GS6 RAM:<br>0: CPU Writes are allowed.<br>1: CPU Writes are blocked.<br>Reset type: SYSRSn |
| 16 | FETCHPROT_GS6 | R/W | 0h | Fetch Protection For GS6 RAM:<br>0: CPU Fetch are allowed.<br>1: CPU Fetch are blocked.<br>Reset type: SYSRSn |
| 15-11 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-262. GSxACCPROT1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 10 | DMAWRPROT_GS5 | R/W | 0h | DMA WR Protection For GS5 RAM:<br>0: DMA Writes are allowed.<br>1: DMA Writes are blocked.<br>Reset type: SYSRSn |
| 9 | CPUWRPROT_GS5 | R/W | 0h | CPU WR Protection For GS5 RAM:<br>0: CPU Writes are allowed.<br>1: CPU Writes are blocked.<br>Reset type: SYSRSn |
| 8 | FETCHPROT_GS5 | R/W | 0h | Fetch Protection For GS5 RAM:<br>0: CPU Fetch are allowed.<br>1: CPU Fetch are blocked.<br>Reset type: SYSRSn |
| 7-3 | RESERVED | R | 0h | Reserved |
| 2 | DMAWRPROT_GS4 | R/W | 0h | DMA WR Protection For GS4 RAM:<br>0: DMA Writes are allowed.<br>1: DMA Writes are blocked.<br>Reset type: SYSRSn |
| 1 | CPUWRPROT_GS4 | R/W | 0h | CPU WR Protection For GS4 RAM:<br>0: CPU Writes are allowed.<br>1: CPU Writes are blocked.<br>Reset type: SYSRSn |
| 0 | FETCHPROT_GS4 | R/W | 0h | Fetch Protection For GS4 RAM:<br>0: CPU Fetch are allowed.<br>1: CPU Fetch are blocked.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.17.21 GSxACCPROT2 Register (Offset = 4Ch) [Reset = 00000000h]

GSxACCPROT2 is shown in Figure 3-236 and described in Table 3-263.

Return to the Summary Table.

Global Shared RAM Config Register 2

**Figure 3-236. GSxACCPROT2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | DMAWRPROT_GS11 | CPUWRPROT_GS11 | FETCHPROT_GS11 |
| R-0h | | | | | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | DMAWRPROT_GS10 | CPUWRPROT_GS10 | FETCHPROT_GS10 |
| R-0h | | | | | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | DMAWRPROT_GS9 | CPUWRPROT_GS9 | FETCHPROT_GS9 |
| R-0h | | | | | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | DMAWRPROT_GS8 | CPUWRPROT_GS8 | FETCHPROT_GS8 |
| R-0h | | | | | R/W-0h | R/W-0h | R/W-0h |

**Table 3-263. GSxACCPROT2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-27 | RESERVED | R | 0h | Reserved |
| 26 | DMAWRPROT_GS11 | R/W | 0h | DMA WR Protection For GS11 RAM:<br>0: DMA Writes are allowed.<br>1: DMA Writes are blocked.<br>Reset type: SYSRSn |
| 25 | CPUWRPROT_GS11 | R/W | 0h | CPU WR Protection For GS11 RAM:<br>0: CPU Writes are allowed.<br>1: CPU Writes are blocked.<br>Reset type: SYSRSn |
| 24 | FETCHPROT_GS11 | R/W | 0h | Fetch Protection For GS11 RAM:<br>0: CPU Fetch are allowed.<br>1: CPU Fetch are blocked.<br>Reset type: SYSRSn |
| 23-19 | RESERVED | R | 0h | Reserved |
| 18 | DMAWRPROT_GS10 | R/W | 0h | DMA WR Protection For GS10 RAM:<br>0: DMA Writes are allowed.<br>1: DMA Writes are blocked.<br>Reset type: SYSRSn |
| 17 | CPUWRPROT_GS10 | R/W | 0h | CPU WR Protection For GS10 RAM:<br>0: CPU Writes are allowed.<br>1: CPU Writes are blocked.<br>Reset type: SYSRSn |
| 16 | FETCHPROT_GS10 | R/W | 0h | Fetch Protection For GS10 RAM:<br>0: CPU Fetch are allowed.<br>1: CPU Fetch are blocked.<br>Reset type: SYSRSn |
| 15-11 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-263. GSxACCPROT2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 10 | DMAWRPROT_GS9 | R/W | 0h | DMA WR Protection For GS9 RAM:<br>0: DMA Writes are allowed.<br>1: DMA Writes are blocked.<br>Reset type: SYSRSn |
| 9 | CPUWRPROT_GS9 | R/W | 0h | CPU WR Protection For GS9 RAM:<br>0: CPU Writes are allowed.<br>1: CPU Writes are blocked.<br>Reset type: SYSRSn |
| 8 | FETCHPROT_GS9 | R/W | 0h | Fetch Protection For GS9 RAM:<br>0: CPU Fetch are allowed.<br>1: CPU Fetch are blocked.<br>Reset type: SYSRSn |
| 7-3 | RESERVED | R | 0h | Reserved |
| 2 | DMAWRPROT_GS8 | R/W | 0h | DMA WR Protection For GS8 RAM:<br>0: DMA Writes are allowed.<br>1: DMA Writes are blocked.<br>Reset type: SYSRSn |
| 1 | CPUWRPROT_GS8 | R/W | 0h | CPU WR Protection For GS8 RAM:<br>0: CPU Writes are allowed.<br>1: CPU Writes are blocked.<br>Reset type: SYSRSn |
| 0 | FETCHPROT_GS8 | R/W | 0h | Fetch Protection For GS8 RAM:<br>0: CPU Fetch are allowed.<br>1: CPU Fetch are blocked.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.17.22 GSxACCPROT3 Register (Offset = 4Eh) [Reset = 00000000h]

GSxACCPROT3 is shown in Figure 3-237 and described in Table 3-264.

Return to the Summary Table.

Global Shared RAM Config Register 3

**Figure 3-237. GSxACCPROT3 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | DMAWRPROT_GS15 | CPUWRPROT_GS15 | FETCHPROT_GS15 |
| R-0h | | | | | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | DMAWRPROT_GS14 | CPUWRPROT_GS14 | FETCHPROT_GS14 |
| R-0h | | | | | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | DMAWRPROT_GS13 | CPUWRPROT_GS13 | FETCHPROT_GS13 |
| R-0h | | | | | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | DMAWRPROT_GS12 | CPUWRPROT_GS12 | FETCHPROT_GS12 |
| R-0h | | | | | R/W-0h | R/W-0h | R/W-0h |

**Table 3-264. GSxACCPROT3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-27 | RESERVED | R | 0h | Reserved |
| 26 | DMAWRPROT_GS15 | R/W | 0h | DMA WR Protection For GS15 RAM:<br>0: DMA Writes are allowed.<br>1: DMA Writes are blocked.<br>Reset type: SYSRSn |
| 25 | CPUWRPROT_GS15 | R/W | 0h | CPU WR Protection For GS15 RAM:<br>0: CPU Writes are allowed.<br>1: CPU Writes are blocked.<br>Reset type: SYSRSn |
| 24 | FETCHPROT_GS15 | R/W | 0h | Fetch Protection For GS15 RAM:<br>0: CPU Fetch are allowed.<br>1: CPU Fetch are blocked.<br>Reset type: SYSRSn |
| 23-19 | RESERVED | R | 0h | Reserved |
| 18 | DMAWRPROT_GS14 | R/W | 0h | DMA WR Protection For GS14 RAM:<br>0: DMA Writes are allowed.<br>1: DMA Writes are blocked.<br>Reset type: SYSRSn |
| 17 | CPUWRPROT_GS14 | R/W | 0h | CPU WR Protection For GS14 RAM:<br>0: CPU Writes are allowed.<br>1: CPU Writes are blocked.<br>Reset type: SYSRSn |
| 16 | FETCHPROT_GS14 | R/W | 0h | Fetch Protection For GS14 RAM:<br>0: CPU Fetch are allowed.<br>1: CPU Fetch are blocked.<br>Reset type: SYSRSn |
| 15-11 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-264. GSxACCPROT3 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 10 | DMAWRPROT_GS13 | R/W | 0h | DMA WR Protection For GS13 RAM:<br>0: DMA Writes are allowed.<br>1: DMA Writes are blocked.<br>Reset type: SYSRSn |
| 9 | CPUWRPROT_GS13 | R/W | 0h | CPU WR Protection For GS13 RAM:<br>0: CPU Writes are allowed.<br>1: CPU Writes are blocked.<br>Reset type: SYSRSn |
| 8 | FETCHPROT_GS13 | R/W | 0h | Fetch Protection For GS13 RAM:<br>0: CPU Fetch are allowed.<br>1: CPU Fetch are blocked.<br>Reset type: SYSRSn |
| 7-3 | RESERVED | R | 0h | Reserved |
| 2 | DMAWRPROT_GS12 | R/W | 0h | DMA WR Protection For GS12 RAM:<br>0: DMA Writes are allowed.<br>1: DMA Writes are blocked.<br>Reset type: SYSRSn |
| 1 | CPUWRPROT_GS12 | R/W | 0h | CPU WR Protection For GS12 RAM:<br>0: CPU Writes are allowed.<br>1: CPU Writes are blocked.<br>Reset type: SYSRSn |
| 0 | FETCHPROT_GS12 | R/W | 0h | Fetch Protection For GS12 RAM:<br>0: CPU Fetch are allowed.<br>1: CPU Fetch are blocked.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.17.23 GSxTEST Register (Offset = 50h) [Reset = 00000000h]

GSxTEST is shown in Figure 3-238 and described in Table 3-265.

Return to the Summary Table.

Global Shared RAM TEST Register

**Figure 3-238. GSxTEST Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| TEST_GS15 | | TEST_GS14 | | TEST_GS13 | | TEST_GS12 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| TEST_GS11 | | TEST_GS10 | | TEST_GS9 | | TEST_GS8 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| TEST_GS7 | | TEST_GS6 | | TEST_GS5 | | TEST_GS4 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TEST_GS3 | | TEST_GS2 | | TEST_GS1 | | TEST_GS0 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 3-265. GSxTEST Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | TEST_GS15 | R/W | 0h | Selects the different modes for GS15 RAM:<br>00: Functional Mode.<br>01: Writes are allowed to data bits only. No write to parity bits.<br>10: Writes are allowed to parity bits only. No write to data bits.<br>11: Functional Mode.<br>Reset type: SYSRSn |
| 29-28 | TEST_GS14 | R/W | 0h | Selects the different modes for GS14 RAM:<br>00: Functional Mode.<br>01: Writes are allowed to data bits only. No write to parity bits.<br>10: Writes are allowed to parity bits only. No write to data bits.<br>11: Functional Mode.<br>Reset type: SYSRSn |
| 27-26 | TEST_GS13 | R/W | 0h | Selects the different modes for GS13 RAM:<br>00: Functional Mode.<br>01: Writes are allowed to data bits only. No write to parity bits.<br>10: Writes are allowed to parity bits only. No write to data bits.<br>11: Functional Mode.<br>Reset type: SYSRSn |
| 25-24 | TEST_GS12 | R/W | 0h | Selects the different modes for GS12 RAM:<br>00: Functional Mode.<br>01: Writes are allowed to data bits only. No write to parity bits.<br>10: Writes are allowed to parity bits only. No write to data bits.<br>11: Functional Mode.<br>Reset type: SYSRSn |
| 23-22 | TEST_GS11 | R/W | 0h | Selects the different modes for GS11 RAM:<br>00: Functional Mode.<br>01: Writes are allowed to data bits only. No write to parity bits.<br>10: Writes are allowed to parity bits only. No write to data bits.<br>11: Functional Mode.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-265. GSxTEST Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 21-20 | TEST_GS10 | R/W | 0h | Selects the different modes for GS10 RAM:<br>00: Functional Mode.<br>01: Writes are allowed to data bits only. No write to parity bits.<br>10: Writes are allowed to parity bits only. No write to data bits.<br>11: Functional Mode.<br>Reset type: SYSRSn |
| 19-18 | TEST_GS9 | R/W | 0h | Selects the different modes for GS9 RAM:<br>00: Functional Mode.<br>01: Writes are allowed to data bits only. No write to parity bits.<br>10: Writes are allowed to parity bits only. No write to data bits.<br>11: Functional Mode.<br>Reset type: SYSRSn |
| 17-16 | TEST_GS8 | R/W | 0h | Selects the different modes for GS8 RAM:<br>00: Functional Mode.<br>01: Writes are allowed to data bits only. No write to parity bits.<br>10: Writes are allowed to parity bits only. No write to data bits.<br>11: Functional Mode.<br>Reset type: SYSRSn |
| 15-14 | TEST_GS7 | R/W | 0h | Selects the different modes for GS7 RAM:<br>00: Functional Mode.<br>01: Writes are allowed to data bits only. No write to parity bits.<br>10: Writes are allowed to parity bits only. No write to data bits.<br>11: Functional Mode.<br>Reset type: SYSRSn |
| 13-12 | TEST_GS6 | R/W | 0h | Selects the different modes for GS6 RAM:<br>00: Functional Mode.<br>01: Writes are allowed to data bits only. No write to parity bits.<br>10: Writes are allowed to parity bits only. No write to data bits.<br>11: Functional Mode.<br>Reset type: SYSRSn |
| 11-10 | TEST_GS5 | R/W | 0h | Selects the different modes for GS5 RAM:<br>00: Functional Mode.<br>01: Writes are allowed to data bits only. No write to parity bits.<br>10: Writes are allowed to parity bits only. No write to data bits.<br>11: Functional Mode.<br>Reset type: SYSRSn |
| 9-8 | TEST_GS4 | R/W | 0h | Selects the different modes for GS4 RAM:<br>00: Functional Mode.<br>01: Writes are allowed to data bits only. No write to parity bits.<br>10: Writes are allowed to parity bits only. No write to data bits.<br>11: Functional Mode.<br>Reset type: SYSRSn |
| 7-6 | TEST_GS3 | R/W | 0h | Selects the different modes for GS3 RAM:<br>00: Functional Mode.<br>01: Writes are allowed to data bits only. No write to parity bits.<br>10: Writes are allowed to parity bits only. No write to data bits.<br>11: Functional Mode.<br>Reset type: SYSRSn |
| 5-4 | TEST_GS2 | R/W | 0h | Selects the different modes for GS2 RAM:<br>00: Functional Mode.<br>01: Writes are allowed to data bits only. No write to parity bits.<br>10: Writes are allowed to parity bits only. No write to data bits.<br>11: Functional Mode.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-265. GSxTEST Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-2 | TEST_GS1 | R/W | 0h | Selects the different modes for GS1 RAM:<br>00: Functional Mode.<br>01: Writes are allowed to data bits only. No write to parity bits.<br>10: Writes are allowed to parity bits only. No write to data bits.<br>11: Functional Mode.<br>Reset type: SYSRSn |
| 1-0 | TEST_GS0 | R/W | 0h | Selects the different modes for GS0 RAM:<br>00: Functional Mode.<br>01: Writes are allowed to data bits only. No write to parity bits.<br>10: Writes are allowed to parity bits only. No write to data bits.<br>11: Functional Mode.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.17.24 GSxINIT Register (Offset = 52h) [Reset = 00000000h]

GSxINIT is shown in Figure 3-239 and described in Table 3-266.

Return to the Summary Table.

Global Shared RAM Init Register

**Figure 3-239. GSxINIT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| INIT_GS15 | INIT_GS14 | INIT_GS13 | INIT_GS12 | INIT_GS11 | INIT_GS10 | INIT_GS9 | INIT_GS8 |
| R-0/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 |
| INIT_GS7 | INIT_GS6 | INIT_GS5 | INIT_GS4 | INIT_GS3 | INIT_GS2 | INIT_GS1 | INIT_GS0 |
| 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 3-266. GSxINIT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15 | INIT_GS15 | R-0/W1S | 0h | RAM Initialization control for GS15 RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |
| 14 | INIT_GS14 | R-0/W1S | 0h | RAM Initialization control for GS14 RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |
| 13 | INIT_GS13 | R-0/W1S | 0h | RAM Initialization control for GS13 RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |
| 12 | INIT_GS12 | R-0/W1S | 0h | RAM Initialization control for GS12 RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |
| 11 | INIT_GS11 | R-0/W1S | 0h | RAM Initialization control for GS11 RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |
| 10 | INIT_GS10 | R-0/W1S | 0h | RAM Initialization control for GS10 RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |
| 9 | INIT_GS9 | R-0/W1S | 0h | RAM Initialization control for GS9 RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-266. GSxINIT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 8 | INIT_GS8 | R-0/W1S | 0h | RAM Initialization control for GS8 RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |
| 7 | INIT_GS7 | R-0/W1S | 0h | RAM Initialization control for GS7 RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |
| 6 | INIT_GS6 | R-0/W1S | 0h | RAM Initialization control for GS6 RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |
| 5 | INIT_GS5 | R-0/W1S | 0h | RAM Initialization control for GS5 RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |
| 4 | INIT_GS4 | R-0/W1S | 0h | RAM Initialization control for GS4 RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |
| 3 | INIT_GS3 | R-0/W1S | 0h | RAM Initialization control for GS3 RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |
| 2 | INIT_GS2 | R-0/W1S | 0h | RAM Initialization control for GS2 RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |
| 1 | INIT_GS1 | R-0/W1S | 0h | RAM Initialization control for GS1 RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |
| 0 | INIT_GS0 | R-0/W1S | 0h | RAM Initialization control for GS0 RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.17.25 GSxINITDONE Register (Offset = 54h) [Reset = 00000000h]

GSxINITDONE is shown in Figure 3-240 and described in Table 3-267.

Return to the Summary Table.

Global Shared RAM InitDone Status Register

**Figure 3-240. GSxINITDONE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| INITDONE_GS15 | INITDONE_GS14 | INITDONE_GS13 | INITDONE_GS12 | INITDONE_GS11 | INITDONE_GS10 | INITDONE_GS9 | INITDONE_GS8 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INITDONE_GS7 | INITDONE_GS6 | INITDONE_GS5 | INITDONE_GS4 | INITDONE_GS3 | INITDONE_GS2 | INITDONE_GS1 | INITDONE_GS0 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 3-267. GSxINITDONE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15 | INITDONE_GS15 | R | 0h | RAM Initialization status for GS15 RAM:<br>0: RAM Initialization is not done.<br>1: RAM Initialization is done.<br>Reset type: SYSRSn |
| 14 | INITDONE_GS14 | R | 0h | RAM Initialization status for GS14 RAM:<br>0: RAM Initialization is not done.<br>1: RAM Initialization is done.<br>Reset type: SYSRSn |
| 13 | INITDONE_GS13 | R | 0h | RAM Initialization status for GS13 RAM:<br>0: RAM Initialization is not done.<br>1: RAM Initialization is done.<br>Reset type: SYSRSn |
| 12 | INITDONE_GS12 | R | 0h | RAM Initialization status for GS12 RAM:<br>0: RAM Initialization is not done.<br>1: RAM Initialization is done.<br>Reset type: SYSRSn |
| 11 | INITDONE_GS11 | R | 0h | RAM Initialization status for GS11 RAM:<br>0: RAM Initialization is not done.<br>1: RAM Initialization is done.<br>Reset type: SYSRSn |
| 10 | INITDONE_GS10 | R | 0h | RAM Initialization status for GS10 RAM:<br>0: RAM Initialization is not done.<br>1: RAM Initialization is done.<br>Reset type: SYSRSn |
| 9 | INITDONE_GS9 | R | 0h | RAM Initialization status for GS9 RAM:<br>0: RAM Initialization is not done.<br>1: RAM Initialization is done.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-267. GSxINITDONE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 8 | INITDONE_GS8 | R | 0h | RAM Initialization status for GS8 RAM:<br>0: RAM Initialization is not done.<br>1: RAM Initialization is done.<br>Reset type: SYSRSn |
| 7 | INITDONE_GS7 | R | 0h | RAM Initialization status for GS7 RAM:<br>0: RAM Initialization is not done.<br>1: RAM Initialization is done.<br>Reset type: SYSRSn |
| 6 | INITDONE_GS6 | R | 0h | RAM Initialization status for GS6 RAM:<br>0: RAM Initialization is not done.<br>1: RAM Initialization is done.<br>Reset type: SYSRSn |
| 5 | INITDONE_GS5 | R | 0h | RAM Initialization status for GS5 RAM:<br>0: RAM Initialization is not done.<br>1: RAM Initialization is done.<br>Reset type: SYSRSn |
| 4 | INITDONE_GS4 | R | 0h | RAM Initialization status for GS4 RAM:<br>0: RAM Initialization is not done.<br>1: RAM Initialization is done.<br>Reset type: SYSRSn |
| 3 | INITDONE_GS3 | R | 0h | RAM Initialization status for GS3 RAM:<br>0: RAM Initialization is not done.<br>1: RAM Initialization is done.<br>Reset type: SYSRSn |
| 2 | INITDONE_GS2 | R | 0h | RAM Initialization status for GS2 RAM:<br>0: RAM Initialization is not done.<br>1: RAM Initialization is done.<br>Reset type: SYSRSn |
| 1 | INITDONE_GS1 | R | 0h | RAM Initialization status for GS1 RAM:<br>0: RAM Initialization is not done.<br>1: RAM Initialization is done.<br>Reset type: SYSRSn |
| 0 | INITDONE_GS0 | R | 0h | RAM Initialization status for GS0 RAM:<br>0: RAM Initialization is not done.<br>1: RAM Initialization is done.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.17.26 MSGxTEST Register (Offset = 70h) [Reset = 00000000h]

MSGxTEST is shown in Figure 3-241 and described in Table 3-268.

Return to the Summary Table.

Message RAM TEST Register

**Figure 3-241. MSGxTEST Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | RESERVED | |
| R-0h | | | | | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | TEST_CLA1TOCPU | | TEST_CPUTOCLA1 | | TEST_CPUTOCPU | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 3-268. MSGxTEST Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-10 | RESERVED | R | 0h | Reserved |
| 9-8 | RESERVED | R/W | 0h | Reserved |
| 7-6 | RESERVED | R/W | 0h | Reserved |
| 5-4 | TEST_CLA1TOCPU | R/W | 0h | Selects the different modes for CLA1TOCPU MSG RAM:<br>00: Functional Mode.<br>01: Writes are allowed to data bits only. No write to parity bits.<br>10: Writes are allowed to parity bits only. No write to data bits.<br>11: Functional Mode.<br>Reset type: SYSRSn |
| 3-2 | TEST_CPUTOCLA1 | R/W | 0h | Selects the different modes for CPUTOCLA1 MSG RAM:<br>00: Functional Mode.<br>01: Writes are allowed to data bits only. No write to parity bits.<br>10: Writes are allowed to parity bits only. No write to data bits.<br>11: Functional Mode.<br>Reset type: SYSRSn |
| 1-0 | TEST_CPUTOCPU | R/W | 0h | Selects the different modes for CPUTOCPU MSG RAM:<br>00: Functional Mode.<br>01: Writes are allowed to data bits only. No write to parity bits.<br>10: Writes are allowed to parity bits only. No write to data bits.<br>11: Functional Mode.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.17.27 MSGxINIT Register (Offset = 72h) [Reset = 00000000h]

MSGxINIT is shown in Figure 3-242 and described in Table 3-269.

Return to the Summary Table.

Message RAM Init Register

**Figure 3-242. MSGxINIT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | RESERVED | RESERVED | INIT_CLA1TOCPU | INIT_CPUTOCLA1 | INIT_CPUTOCPU |
| R-0h | | | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h |

**Table 3-269. MSGxINIT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-5 | RESERVED | R | 0h | Reserved |
| 4 | RESERVED | R-0/W1S | 0h | Reserved |
| 3 | RESERVED | R-0/W1S | 0h | Reserved |
| 2 | INIT_CLA1TOCPU | R-0/W1S | 0h | RAM Initialization control for CLA1TOCPU MSG RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |
| 1 | INIT_CPUTOCLA1 | R-0/W1S | 0h | RAM Initialization control for CPUTOCLA1 MSG RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |
| 0 | INIT_CPUTOCPU | R-0/W1S | 0h | RAM Initialization control for CPUTOCPU MSG RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.17.28 MSGxINITDONE Register (Offset = 74h) [Reset = 00000000h]

MSGxINITDONE is shown in Figure 3-243 and described in Table 3-270.

Return to the Summary Table.

Message RAM InitDone Status Register

**Figure 3-243. MSGxINITDONE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | RESERVED | RESERVED | INITDONE_CLA1TOCPU | INITDONE_CPUTOCLA1 | INITDONE_CPUTOCPU |
| R-0h | | | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 3-270. MSGxINITDONE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-5 | RESERVED | R | 0h | Reserved |
| 4 | RESERVED | R | 0h | Reserved |
| 3 | RESERVED | R | 0h | Reserved |
| 2 | INITDONE_CLA1TOCPU | R | 0h | RAM Initialization status for CLA1TOCPU MSG RAM:<br>0: RAM Initialization is not done.<br>1: RAM Initialization is done.<br>Reset type: SYSRSn |
| 1 | INITDONE_CPUTOCLA1 | R | 0h | RAM Initialization status for CPUTOCLA1 MSG RAM:<br>0: RAM Initialization is not done.<br>1: RAM Initialization is done.<br>Reset type: SYSRSn |
| 0 | INITDONE_CPUTOCPU | R | 0h | RAM Initialization status for CPUTOCPU MSG RAM:<br>0: RAM Initialization is not done.<br>1: RAM Initialization is done.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.18 ACCESS_PROTECTION_REGS Registers

Table 3-271 lists the memory-mapped registers for the ACCESS_PROTECTION_REGS registers. All register offset addresses not listed in Table 3-271 should be considered as reserved locations and the register contents should not be modified.

**Table 3-271. ACCESS_PROTECTION_REGS Registers**

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

Complex bit access types are encoded to fit into small table cells. Table 3-272 shows the codes that are used for access types in this section.

**Table 3-272. ACCESS_PROTECTION_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| R-0 | R<br>-0 | Read<br>Returns 0s |
| Write Type | | |
| W | W | Write |
| W1S | W<br>1S | Write<br>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. |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-272. ACCESS_PROTECTION_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. |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.18.1 NMAVFLG Register (Offset = 0h) [Reset = 00000000h]

NMAVFLG is shown in Figure 3-244 and described in Table 3-273.

Return to the Summary Table.

Non-Master Access Violation Flag Register

**Figure 3-244. NMAVFLG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | RESERVED | RESERVED |
| R-0h | | | | | | R-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | CLA1FETCH | CLA1WRITE | CLA1READ | DMAWRITE | CPUFETCH | CPUWRITE | CPUREAD |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 3-273. NMAVFLG Register Field Descriptions**

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



Copyright © 2024 Texas Instruments Incorporated

**Table 3-273. NMAVFLG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | CPUREAD | R | 0h | Non Master CPU Read Access Violation Flag:<br>0: No violation.<br>1: Access violation occured.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.18.2 NMAVSET Register (Offset = 2h) [Reset = 00000000h]

NMAVSET is shown in Figure 3-245 and described in Table 3-274.

Return to the Summary Table.

Non-Master Access Violation Flag Set Register

**Figure 3-245. NMAVSET Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | RESERVED | RESERVED |
| R-0h | | | | | | R-0/W1S-0h | R-0/W1S-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | CLA1FETCH | CLA1WRITE | CLA1READ | DMAWRITE | CPUFETCH | CPUWRITE | CPUREAD |
| R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h |

**Table 3-274. NMAVSET Register Field Descriptions**

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



Copyright © 2024 Texas Instruments Incorporated

**Table 3-274. NMAVSET Register Field Descriptions (continued)**

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



Copyright © 2024 Texas Instruments Incorporated

### 3.17.18.3 NMAVCLR Register (Offset = 4h) [Reset = 00000000h]

NMAVCLR is shown in Figure 3-246 and described in Table 3-275.

Return to the Summary Table.

Non-Master Access Violation Flag Clear Register

**Figure 3-246. NMAVCLR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | RESERVED | RESERVED |
| R-0h | | | | | | R-0/W1S-0h | R-0/W1S-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | CLA1FETCH | CLA1WRITE | CLA1READ | DMAWRITE | CPUFETCH | CPUWRITE | CPUREAD |
| R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h |

**Table 3-275. NMAVCLR Register Field Descriptions**

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



Copyright © 2024 Texas Instruments Incorporated

**Table 3-275. NMAVCLR Register Field Descriptions (continued)**

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



Copyright © 2024 Texas Instruments Incorporated

### 3.17.18.4 NMAVINTEN Register (Offset = 6h) [Reset = 00000000h]

NMAVINTEN is shown in Figure 3-247 and described in Table 3-276.

Return to the Summary Table.

Non-Master Access Violation Interrupt Enable Register

**Figure 3-247. NMAVINTEN Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | RESERVED | RESERVED |
| R-0h | | | | | | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | CLA1FETCH | CLA1WRITE | CLA1READ | DMAWRITE | CPUFETCH | CPUWRITE | CPUREAD |
| 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 3-276. NMAVINTEN Register Field Descriptions**

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



Copyright © 2024 Texas Instruments Incorporated

### 3.17.18.5 NMCPURDAVADDR Register (Offset = 8h) [Reset = 00000000h]

NMCPURDAVADDR is shown in Figure 3-248 and described in Table 3-277.

Return to the Summary Table.

Non-Master CPU Read Access Violation Address

**Figure 3-248. NMCPURDAVADDR Register**

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

**Table 3-277. NMCPURDAVADDR Register Field Descriptions**

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



Copyright © 2024 Texas Instruments Incorporated

### 3.17.18.6 NMCPUWRAVADDR Register (Offset = Ah) [Reset = 00000000h]

NMCPUWRAVADDR is shown in Figure 3-249 and described in Table 3-278.

Return to the Summary Table.

Non-Master CPU Write Access Violation Address

**Figure 3-249. NMCPUWRAVADDR Register**

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

**Table 3-278. NMCPUWRAVADDR Register Field Descriptions**

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



Copyright © 2024 Texas Instruments Incorporated

### 3.17.18.7 NMCPUFAVADDR Register (Offset = Ch) [Reset = 00000000h]

NMCPUFAVADDR is shown in Figure 3-250 and described in Table 3-279.

Return to the Summary Table.

Non-Master CPU Fetch Access Violation Address

**Figure 3-250. NMCPUFAVADDR Register**

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

**Table 3-279. NMCPUFAVADDR Register Field Descriptions**

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



Copyright © 2024 Texas Instruments Incorporated

### 3.17.18.8 NMDMAWRAVADDR Register (Offset = Eh) [Reset = 00000000h]

NMDMAWRAVADDR is shown in Figure 3-251 and described in Table 3-280.

Return to the Summary Table.

Non-Master DMA Write Access Violation Address

**Figure 3-251. NMDMAWRAVADDR Register**

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

**Table 3-280. NMDMAWRAVADDR Register Field Descriptions**

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



Copyright © 2024 Texas Instruments Incorporated

### 3.17.18.9 NMCLA1RDAVADDR Register (Offset = 10h) [Reset = 00000000h]

NMCLA1RDAVADDR is shown in Figure 3-252 and described in Table 3-281.

Return to the Summary Table.

Non-Master CLA1 Read Access Violation Address

**Figure 3-252. NMCLA1RDAVADDR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NMCLA1RDAVADDR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 3-281. NMCLA1RDAVADDR Register Field Descriptions**

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



Copyright © 2024 Texas Instruments Incorporated

### 3.17.18.10 NMCLA1WRAVADDR Register (Offset = 12h) [Reset = 00000000h]

NMCLA1WRAVADDR is shown in Figure 3-253 and described in Table 3-282.

Return to the Summary Table.

Non-Master CLA1 Write Access Violation Address

**Figure 3-253. NMCLA1WRAVADDR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NMCLA1WRAVADDR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 3-282. NMCLA1WRAVADDR Register Field Descriptions**

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



Copyright © 2024 Texas Instruments Incorporated

### 3.17.18.11 NMCLA1FAVADDR Register (Offset = 14h) [Reset = 00000000h]

NMCLA1FAVADDR is shown in Figure 3-254 and described in Table 3-283.

Return to the Summary Table.

Non-Master CLA1 Fetch Access Violation Address

**Figure 3-254. NMCLA1FAVADDR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NMCLA1FAVADDR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 3-283. NMCLA1FAVADDR Register Field Descriptions**

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



Copyright © 2024 Texas Instruments Incorporated

### 3.17.18.12 MAVFLG Register (Offset = 20h) [Reset = 00000000h]

MAVFLG is shown in Figure 3-255 and described in Table 3-284.

Return to the Summary Table.

Master Access Violation Flag Register

**Figure 3-255. MAVFLG Register**

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

**Table 3-284. MAVFLG Register Field Descriptions**

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



Copyright © 2024 Texas Instruments Incorporated

### 3.17.18.13 MAVSET Register (Offset = 22h) [Reset = 00000000h]

MAVSET is shown in Figure 3-256 and described in Table 3-285.

Return to the Summary Table.

Master Access Violation Flag Set Register

**Figure 3-256. MAVSET Register**

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

**Table 3-285. MAVSET Register Field Descriptions**

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



Copyright © 2024 Texas Instruments Incorporated

### 3.17.18.14 MAVCLR Register (Offset = 24h) [Reset = 00000000h]

MAVCLR is shown in Figure 3-257 and described in Table 3-286.

Return to the Summary Table.

Master Access Violation Flag Clear Register

**Figure 3-257. MAVCLR Register**

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

**Table 3-286. MAVCLR Register Field Descriptions**

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



Copyright © 2024 Texas Instruments Incorporated

### 3.17.18.15 MAVINTEN Register (Offset = 26h) [Reset = 00000000h]

MAVINTEN is shown in Figure 3-258 and described in Table 3-287.

Return to the Summary Table.

Master Access Violation Interrupt Enable Register

**Figure 3-258. MAVINTEN Register**

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

**Table 3-287. MAVINTEN Register Field Descriptions**

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



Copyright © 2024 Texas Instruments Incorporated

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

MCPUFAVADDR is shown in Figure 3-259 and described in Table 3-288.

Return to the Summary Table.

Master CPU Fetch Access Violation Address

**Figure 3-259. 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 3-288. 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 vaiolation occurred.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.18.17 MCPUWRAVADDR Register (Offset = 2Ah) [Reset = 00000000h]

MCPUWRAVADDR is shown in Figure 3-260 and described in Table 3-289.

Return to the Summary Table.

Master CPU Write Access Violation Address

**Figure 3-260. 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 3-289. 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 vaiolation occurred.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.18.18 MDMAWRAVADDR Register (Offset = 2Ch) [Reset = 00000000h]

MDMAWRAVADDR is shown in Figure 3-261 and described in Table 3-290.

Return to the Summary Table.

Master DMA Write Access Violation Address

**Figure 3-261. MDMAWRAVADDR Register**

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

**Table 3-290. MDMAWRAVADDR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | MDMAWRAVADDR | R | 0h | This register captures the address location for which master DMA write access vaiolation occurred.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.19 MEMORY_ERROR_REGS Registers

Table 3-291 lists the memory-mapped registers for the MEMORY_ERROR_REGS registers. All register offset addresses not listed in Table 3-291 should be considered as reserved locations and the register contents should not be modified.

**Table 3-291. MEMORY_ERROR_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | UCERRFLG | Uncorrectable Error Flag Register | | Go |
| 2h | UCERRSET | Uncorrectable Error Flag Set Register | EALLOW | Go |
| 4h | UCERRCLR | Uncorrectable Error Flag Clear Register | EALLOW | Go |
| 6h | UCCPUREADDR | Uncorrectable CPU Read Error Address | | Go |
| 8h | UCDMAREADDR | Uncorrectable DMA Read Error Address | | Go |
| Ah | UCCLA1READDR | Uncorrectable CLA1 Read Error Address | | Go |
| 20h | CERRFLG | Correctable Error Flag Register | | Go |
| 22h | CERRSET | Correctable Error Flag Set Register | EALLOW | Go |
| 24h | CERRCLR | Correctable Error Flag Clear Register | EALLOW | Go |
| 26h | CCPUREADDR | Correctable CPU Read Error Address | | Go |
| 2Eh | CERRCNT | Correctable Error Count Register | | Go |
| 30h | CERRTHRES | Correctable Error Threshold Value Register | EALLOW | Go |
| 32h | CEINTFLG | Correctable Error Interrupt Flag Status Register | | Go |
| 34h | CEINTCLR | Correctable Error Interrupt Flag Clear Register | EALLOW | Go |
| 36h | CEINTSET | Correctable Error Interrupt Flag Set Register | EALLOW | Go |
| 38h | CEINTEN | Correctable Error Interrupt Enable Register | EALLOW | Go |

Complex bit access types are encoded to fit into small table cells. Table 3-292 shows the codes that are used for access types in this section.

**Table 3-292. MEMORY_ERROR_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| R-0 | R<br>-0 | Read<br>Returns 0s |
| Write Type | | |
| W | W | Write |
| W1S | W<br>1S | Write<br>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. |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.19.1 UCERRFLG Register (Offset = 0h) [Reset = 00000000h]

UCERRFLG is shown in Figure 3-262 and described in Table 3-293.

Return to the Summary Table.

Uncorrectable Error Flag Register

**Figure 3-262. UCERRFLG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | RESERVED | CLA1RDERR | DMARDERR | CPURDERR |
| R-0h | | | | R-0h | R-0h | R-0h | R-0h |

**Table 3-293. UCERRFLG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-4 | RESERVED | R | 0h | Reserved |
| 3 | RESERVED | R | 0h | Reserved |
| 2 | CLA1RDERR | R | 0h | CLA1 Uncorrectable Read Error Flag<br>0: No Error.<br>1: Uncorrectable error occurred during CLA1 read.<br>Reset type: SYSRSn |
| 1 | DMARDERR | R | 0h | DMA Uncorrectable Read Error Flag<br>0: No Error.<br>1: Uncorrectable error occurred during DMA read.<br>Reset type: SYSRSn |
| 0 | CPURDERR | R | 0h | CPU Uncorrectable Read Error Flag<br>0: No Error.<br>1: Uncorrectable error occurred during CPU read.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.19.2 UCERRSET Register (Offset = 2h) [Reset = 00000000h]

UCERRSET is shown in Figure 3-263 and described in Table 3-294.

Return to the Summary Table.

Uncorrectable Error Flag Set Register

**Figure 3-263. UCERRSET Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | RESERVED | CLA1RDERR | DMARDERR | CPURDERR |
| R-0h | | | | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h |

**Table 3-294. UCERRSET Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-4 | RESERVED | R | 0h | Reserved |
| 3 | RESERVED | R-0/W1S | 0h | Reserved |
| 2 | CLA1RDERR | R-0/W1S | 0h | 0: No action.<br>1: CLA1 Read Error Flag in UCERRFLG register will be set and interrupt will be generated if enabled..<br>Reset type: SYSRSn |
| 1 | DMARDERR | R-0/W1S | 0h | 0: No action.<br>1: DMA Read Error Flag in UCERRFLG register will be set and interrupt will be generated if enabled..<br>Reset type: SYSRSn |
| 0 | CPURDERR | R-0/W1S | 0h | 0: No action.<br>1: CPU Read Error Flag in UCERRFLG register will be set and interrupt will be generated if enabled..<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.19.3 UCERRCLR Register (Offset = 4h) [Reset = 00000000h]

UCERRCLR is shown in Figure 3-264 and described in Table 3-295.

Return to the Summary Table.

Uncorrectable Error Flag Clear Register

**Figure 3-264. UCERRCLR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | RESERVED | CLA1RDERR | DMARDERR | CPURDERR |
| R-0h | | | | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h |

**Table 3-295. UCERRCLR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-4 | RESERVED | R | 0h | Reserved |
| 3 | RESERVED | R-0/W1S | 0h | Reserved |
| 2 | CLA1RDERR | R-0/W1S | 0h | 0: No action.<br>1: CLA1 Read Error Flag in UCERRFLG register will be cleared.<br>Reset type: SYSRSn |
| 1 | DMARDERR | R-0/W1S | 0h | 0: No action.<br>1: DMA Read Error Flag in UCERRFLG register will be cleared .<br>Reset type: SYSRSn |
| 0 | CPURDERR | R-0/W1S | 0h | 0: No action.<br>1: CPU Read Error Flag in UCERRFLG register will be cleared.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.19.4 UCCPUREADDR Register (Offset = 6h) [Reset = 00000000h]

UCCPUREADDR is shown in Figure 3-265 and described in Table 3-296.

Return to the Summary Table.

Uncorrectable CPU Read Error Address

**Figure 3-265. 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 3-296. 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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.19.5 UCDMAREADDR Register (Offset = 8h) [Reset = 00000000h]

UCDMAREADDR is shown in Figure 3-266 and described in Table 3-297.

Return to the Summary Table.

Uncorrectable DMA Read Error Address

**Figure 3-266. UCDMAREADDR Register**

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

**Table 3-297. UCDMAREADDR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | UCDMAREADDR | R | 0h | This register captures the address location for which DMA read access resulted in uncorrectable ECC/Parity error.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.19.6 UCCLA1READDR Register (Offset = Ah) [Reset = 00000000h]

UCCLA1READDR is shown in Figure 3-267 and described in Table 3-298.

Return to the Summary Table.

Uncorrectable CLA1 Read Error Address

**Figure 3-267. UCCLA1READDR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UCCLA1READDR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 3-298. UCCLA1READDR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | UCCLA1READDR | R | 0h | This register captures the address location for which CLA1 read/ fetch access resulted in uncorrectable ECC/Parity error.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.19.7 CERRFLG Register (Offset = 20h) [Reset = 00000000h]

CERRFLG is shown in Figure 3-268 and described in Table 3-299.

Return to the Summary Table.

Correctable Error Flag Register

**Figure 3-268. CERRFLG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | RESERVED | CLA1RDERR | DMARDERR | CPURDERR |
| R-0h | | | | R-0h | R-0h | R-0h | R-0h |

**Table 3-299. CERRFLG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-4 | RESERVED | R | 0h | Reserved |
| 3 | RESERVED | R | 0h | Reserved |
| 2 | CLA1RDERR | R | 0h | CLA1 Correctable Read Error Flag<br>0: No Error.<br>1: Correctable error occurred during CLA1 read.<br>Reset type: SYSRSn |
| 1 | DMARDERR | R | 0h | DMA Correctable Read Error Flag<br>0: No Error.<br>1: Correctable error occurred during DMA read.<br>Reset type: SYSRSn |
| 0 | CPURDERR | R | 0h | CPU Correctable Read Error Flag<br>0: No Error.<br>1: Correctable error occurred during CPU read.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.19.8 CERRSET Register (Offset = 22h) [Reset = 00000000h]

CERRSET is shown in Figure 3-269 and described in Table 3-300.

Return to the Summary Table.

Correctable Error Flag Set Register

**Figure 3-269. CERRSET Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | RESERVED | CLA1RDERR | DMARDERR | CPURDERR |
| R-0h | | | | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h |

**Table 3-300. CERRSET Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-4 | RESERVED | R | 0h | Reserved |
| 3 | RESERVED | R-0/W1S | 0h | Reserved |
| 2 | CLA1RDERR | R-0/W1S | 0h | 0: No action.<br>1: CLA1 Read Error Flag in CERRFLG register will be set and interrupt will be generated if enabled..<br>Reset type: SYSRSn |
| 1 | DMARDERR | R-0/W1S | 0h | 0: No action.<br>1: DMA Read Error Flag in CERRFLG register will be set and interrupt will be generated if enabled..<br>Reset type: SYSRSn |
| 0 | CPURDERR | R-0/W1S | 0h | 0: No action.<br>1: CPU Read Error Flag in CERRFLG register will be set and interrupt will be generated if enabled..<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.19.9 CERRCLR Register (Offset = 24h) [Reset = 00000000h]

CERRCLR is shown in Figure 3-270 and described in Table 3-301.

Return to the Summary Table.

Correctable Error Flag Clear Register

**Figure 3-270. CERRCLR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | RESERVED | CLA1RDERR | DMARDERR | CPURDERR |
| R-0h | | | | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h |

**Table 3-301. CERRCLR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-4 | RESERVED | R | 0h | Reserved |
| 3 | RESERVED | R-0/W1S | 0h | Reserved |
| 2 | CLA1RDERR | R-0/W1S | 0h | 0: No action.<br>1: CLA1 Read Error Flag in CERRFLG register will be cleared.<br>Reset type: SYSRSn |
| 1 | DMARDERR | R-0/W1S | 0h | 0: No action.<br>1: DMA Read Error Flag in CERRFLG register will be cleared .<br>Reset type: SYSRSn |
| 0 | CPURDERR | R-0/W1S | 0h | 0: No action.<br>1: CPU Read Error Flag in CERRFLG register will be cleared.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.19.10 CCPUREADDR Register (Offset = 26h) [Reset = 00000000h]

CCPUREADDR is shown in Figure 3-271 and described in Table 3-302.

Return to the Summary Table.

Correctable CPU Read Error Address

**Figure 3-271. 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 3-302. 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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.19.11 CERRCNT Register (Offset = 2Eh) [Reset = 00000000h]

CERRCNT is shown in Figure 3-272 and described in Table 3-303.

Return to the Summary Table.

Correctable Error Count Register

**Figure 3-272. 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | | CERRCNT | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | R-0h | | | | | | | | | | | | | | | |

**Table 3-303. CERRCNT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-0 | CERRCNT | R | 0h | This register holds the count of how many times correctable error occurred.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.19.12 CERRTHRES Register (Offset = 30h) [Reset = 00000000h]

CERRTHRES is shown in Figure 3-273 and described in Table 3-304.

Return to the Summary Table.

Correctable Error Threshold Value Register

**Figure 3-273. 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 3-304. 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 or equal to than value configured in this register, corretable interrupt gets generated, if enabled.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.19.13 CEINTFLG Register (Offset = 32h) [Reset = 00000000h]

CEINTFLG is shown in Figure 3-274 and described in Table 3-305.

Return to the Summary Table.

Correctable Error Interrupt Flag Status Register

**Figure 3-274. 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 3-305. 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<br>0: Total correctable errors < Threshold value configured in CERRTHRES register.<br>1: Total correctable errors >= Threshold value configured in CERRTHRES register.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.19.14 CEINTCLR Register (Offset = 34h) [Reset = 00000000h]

CEINTCLR is shown in Figure 3-275 and described in Table 3-306.

Return to the Summary Table.

Correctable Error Interrupt Flag Clear Register

**Figure 3-275. 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 3-306. 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.<br>1: Total corrected error count exceeded flag in CEINTFLG register will be cleared.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.19.15 CEINTSET Register (Offset = 36h) [Reset = 00000000h]

CEINTSET is shown in Figure 3-276 and described in Table 3-307.

Return to the Summary Table.

Correctable Error Interrupt Flag Set Register

**Figure 3-276. 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 3-307. 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.<br>1: Total corrected error count exceeded flag in CEINTFLG register will be set and interrupt will be generated if enabled.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.19.16 CEINTEN Register (Offset = 38h) [Reset = 00000000h]

CEINTEN is shown in Figure 3-277 and described in Table 3-308.

Return to the Summary Table.

Correctable Error Interrupt Enable Register

**Figure 3-277. 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 3-308. 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.<br>1: Correctable Error Interrupt is enabled.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.20 ROM_WAIT_STATE_REGS Registers

Table 3-309 lists the memory-mapped registers for the ROM_WAIT_STATE_REGS registers. All register offset addresses not listed in Table 3-309 should be considered as reserved locations and the register contents should not be modified.

**Table 3-309. ROM_WAIT_STATE_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | ROMWAITSTATE | ROM Wait State Configuration Register | EALLOW | Go |

Complex bit access types are encoded to fit into small table cells. Table 3-310 shows the codes that are used for access types in this section.

**Table 3-310. ROM_WAIT_STATE_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. |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.20.1 ROMWAITSTATE Register (Offset = 0h) [Reset = 00000000h]

ROMWAITSTATE is shown in Figure 3-278 and described in Table 3-311.

Return to the Summary Table.

ROM Wait State Configuration Register

**Figure 3-278. ROMWAITSTATE Register**

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

**Table 3-311. ROMWAITSTATE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-1 | RESERVED | R | 0h | Reserved |
| 0 | WSDISABLE | R/W | 0h | 0: C28x ROM Wait State is enabled. C28x CPU accesses to ROM are are 1-wait.<br>1: C28x ROM Wait State is disabled. C28x CPU accesses to ROM are 0-wait.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.21 FLASH_CTRL_REGS Registers

Table 3-312 lists the memory-mapped registers for the FLASH_CTRL_REGS registers. All register offset addresses not listed in Table 3-312 should be considered as reserved locations and the register contents should not be modified.

**Table 3-312. FLASH_CTRL_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | FRDCNTL | Flash Read Control Register | EALLOW | Go |
| 1Eh | FBAC | Flash Bank Access Control Register | EALLOW | Go |
| 20h | FBFALLBACK | Flash Bank Fallback Power Register | EALLOW | Go |
| 22h | FBPRDY | Flash Bank Pump Ready Register | EALLOW | Go |
| 24h | FPAC1 | Flash Pump Access Control Register 1 | EALLOW | Go |
| 2Ah | FMSTAT | Flash Module Status Register | EALLOW | Go |
| 180h | FRD_INTF_CTRL | Flash Read Interface Control Register | EALLOW | Go |

Complex bit access types are encoded to fit into small table cells. Table 3-313 shows the codes that are used for access types in this section.

**Table 3-313. 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. |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.21.1 FRDCNTL Register (Offset = 0h) [Reset = 00000F00h]

FRDCNTL is shown in Figure 3-279 and described in Table 3-314.

Return to the Summary Table.

Flash Read Control Register

**Figure 3-279. FRDCNTL Register**

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

**Table 3-314. FRDCNTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-12 | RESERVED | R | 0h | Reserved |
| 11-8 | RWAIT | R/W | Fh | Random read waitstate<br>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.<br>Note: The required wait states for each SYSCLK frequency can be found in the device data manual.<br>Reset type: SYSRSn |
| 7-0 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.21.2 FBAC Register (Offset = 1Eh) [Reset = 0000000Fh]

FBAC is shown in Figure 3-280 and described in Table 3-315.

Return to the Summary Table.

Flash Bank Access Control Register

**Figure 3-280. FBAC Register**

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

**Table 3-315. FBAC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-8 | RESERVED | R/W | 0h | Reserved |
| 7-0 | VREADST | R/W | Fh | This bit-field ensures that the requisite delay is introduced for the flash pump/bank to come out of low-power mode, so that the flash/OTP array is ready for CPU accesses.<br>Recommended configuration:<br>0xF (reset value) - If SYSCLK=10MHz (INTOSC)<br>0x14 - If SYSCLK > 10MHz<br>Before entering any low-power mode for the flash bank/pump, this bit must be configured as described in the 'Flash and OTP Memory' chapter of the TRM. Applications typically use the flash bank/pump low-power modes to reduce power for following reasons:<br>(i) during the device-level low-power modes such as IDLE/ STANDBY/HALT<br>(ii) while running code off RAM after powering down the flash.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.21.3 FBFALLBACK Register (Offset = 20h) [Reset = 00000000h]

FBFALLBACK is shown in Figure 3-281 and described in Table 3-316.

Return to the Summary Table.

Flash Bank Fallback Power Register

**Figure 3-281. FBFALLBACK Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | BNKPWR0 | |
| R-0h | | | | | | R/W-0h | |

**Table 3-316. FBFALLBACK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-2 | RESERVED | R | 0h | Reserved |
| 1-0 | BNKPWR0 | R/W | 0h | Bank Power Mode Control<br>00 Sleep (Sense amplifiers and sense reference disabled)<br>01 Standby (Sense amplifiers disabled, but sense reference enabled)<br>10 Reserved<br>11 Active (Both sense amplifiers and sense reference enabled)<br>Note: If the bank and pump are not in active mode and an access is made, the value of this register is automatically changed to active.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.21.4 FBPRDY Register (Offset = 22h) [Reset = 00000000h]

FBPRDY is shown in Figure 3-282 and described in Table 3-317.

Return to the Summary Table.

Flash Bank Pump Ready Register

**Figure 3-282. FBPRDY Register**

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

**Table 3-317. FBPRDY Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15 | PUMPRDY | R | 0h | Pump Ready. This is a read-only bit which allows software to determine if the pump is ready for flash access before attempting the actual access. If an access is made to a bank when the pump is not ready, wait states are asserted until it becomes ready.<br>0 Pump is not ready.<br>1 Pump is ready, in active power state.<br>Reset type: SYSRSn |
| 14-1 | RESERVED | R | 0h | Reserved |
| 0 | BANKRDY | R | 0h | Bank Ready. This is a read-only register which allows software to determine if the bank is ready for Flash access before the access is attempted.<br>Note: The user should wait for both the pump and the bank to be ready before attempting an access.<br>0 Bank is not ready.<br>1 Bank is in active power mode and is ready for access.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.21.5 FPAC1 Register (Offset = 24h) [Reset = 08600000h]

FPAC1 is shown in Figure 3-283 and described in Table 3-318.

Return to the Summary Table.

Flash Pump Access Control Register 1

**Figure 3-283. FPAC1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | PSLEEP | | | |
| R-0h | | | | R/W-860h | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| PSLEEP | | | | | | | |
| R/W-860h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | | PMPPWR |
| R-0h | | | | | | | R/W-0h |

**Table 3-318. FPAC1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-28 | RESERVED | R | 0h | Reserved |
| 27-16 | PSLEEP | R/W | 860h | Pump sleep. These bits contain the starting count value for the charge pump sleep down counter. While the charge pump is in sleep mode, the power mode management logic holds the charge pump sleep counter at this value. When the charge pump exits sleep power mode, the down counter delays from 0 to PSLEEP prescaled SYSCLK clock cycles before putting the charge pump into active power mode.<br>Note: The pump sleep down counter uses a prescaled clock which is divided by 2 of input SYSCLK. The configured PSLEEP value should yield a delay of at least 20 microseconds for the pump to go to active mode.<br>Reset type: SYSRSn |
| 15-1 | RESERVED | R | 0h | Reserved |
| 0 | PMPPWR | R/W | 0h | Flash Charge Pump Control Power Mode.<br>Configures the power mode of the charge pump.<br>0 Sleep (all pump circuits disabled)<br>1 Active (all pump circuits active)<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.21.6 FMSTAT Register (Offset = 2Ah) [Reset = 00000000h]

FMSTAT is shown in Figure 3-284 and described in Table 3-319.

Return to the Summary Table.

Flash Module Status Register

**Figure 3-284. FMSTAT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | RESERVED | RESERVED |
| R-0h | | | | | | R-0h | R-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | RESERVED | RESERVED | PGV | RESERVED | EV | RESERVED | BUSY |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ERS | PGM | INVDAT | CSTAT | VOLTSTAT | ESUSP | PSUSP | RESERVED |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 3-319. FMSTAT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-18 | RESERVED | R | 0h | Reserved |
| 17 | RESERVED | R | 0h | Reserved |
| 16 | RESERVED | R | 0h | Reserved |
| 15 | RESERVED | R | 0h | Reserved |
| 14 | RESERVED | R | 0h | Reserved |
| 13 | RESERVED | R | 0h | Reserved |
| 12 | PGV | R | 0h | Program verify When set, indicates that a word is not successfully programmed after the maximum allowed number of program pulses are given for program operation.<br>Reset type: SYSRSn |
| 11 | RESERVED | R | 0h | Reserved |
| 10 | EV | R | 0h | Erase verify When set, indicates that a sector is not successfully erased after the maximum allowed number of erase pulses are given for erase operation.<br>Reset type: SYSRSn |
| 9 | RESERVED | R | 0h | Reserved |
| 8 | BUSY | R | 0h | When set, this bit indicates that a program, erase, or suspend operation is being processed.<br>Reset type: SYSRSn |
| 7 | ERS | R | 0h | Erase Active. When set, this bit indicates that the flash module is actively performing an erase operation. This bit is set when erasing starts and is cleared when erasing is complete. It is also cleared when the erase is suspended and set when the erase resumes.<br>Reset type: SYSRSn |
| 6 | PGM | R | 0h | Program Active. When set, this bit indicates that the flash module is currently performing a program operation. This bit is set when programming starts and is cleared when programming is complete. It is also cleared when programming is suspended and set when programming is resumed.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-319. FMSTAT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 5 | INVDAT | R | 0h | Invalid Data. When set, this bit indicates that the user attempted to program a '1' where a '0' was already present.<br>Reset type: SYSRSn |
| 4 | CSTAT | R | 0h | Command Status. Once the FSM starts any failure will set this bit. When set, this bit informs the host that the program, erase, or validate sector command failed and the command was stopped. This bit is cleared by the Clear Status command. For some errors, this will be the only indication of an FSM error because the cause does not fall within the other error bit types.<br>Reset type: SYSRSn |
| 3 | VOLTSTAT | R | 0h | Core Voltage Status. When set, this bit indicates that the core voltage generator of the pump power upply dipped below the lower limit allowable during a program or erase operation.<br>Reset type: SYSRSn |
| 2 | ESUSP | R | 0h | When set, this bit indicates that the flash module has received and processed an erase suspend operation. This bit remains set until the erase resume command has been issued or until the Clear_More command is run.<br>Reset type: SYSRSn |
| 1 | PSUSP | R | 0h | When set, this bit indicates that the flash module has received and processed a program suspend operation. This bit remains set until the program resume command has been issued or until the Clear_More command is run.<br>Reset type: SYSRSn |
| 0 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.21.7 FRD_INTF_CTRL Register (Offset = 180h) [Reset = 00000000h]

FRD_INTF_CTRL is shown in Figure 3-285 and described in Table 3-320.

Return to the Summary Table.

Flash Read Interface Control Register

**Figure 3-285. FRD_INTF_CTRL Register**

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

**Table 3-320. 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.<br>0 A value of 0 disables the data cache.<br>1 A value of 1 enables the data cache.<br>Reset type: SYSRSn |
| 0 | PREFETCH_EN | R/W | 0h | Prefetch enable.<br>0 A value of 0 disables prefetch mechanism.<br>1 A value of 1 enables pre-fetch mechanism.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.22 FLASH_ECC_REGS Registers

Table 3-321 lists the memory-mapped registers for the FLASH_ECC_REGS registers. All register offset addresses not listed in Table 3-321 should be considered as reserved locations and the register contents should not be modified.

**Table 3-321. FLASH_ECC_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | ECC_ENABLE | ECC Enable | EALLOW | Go |
| 2h | SINGLE_ERR_ADDR_LOW | Single Error Address Low | EALLOW | Go |
| 4h | SINGLE_ERR_ADDR_HIGH | Single Error Address High | EALLOW | Go |
| 6h | UNC_ERR_ADDR_LOW | Uncorrectable Error Address Low | EALLOW | Go |
| 8h | UNC_ERR_ADDR_HIGH | Uncorrectable Error Address High | EALLOW | Go |
| Ah | ERR_STATUS | Error Status | EALLOW | Go |
| Ch | ERR_POS | Error Position | EALLOW | Go |
| Eh | ERR_STATUS_CLR | Error Status Clear | EALLOW | Go |
| 10h | ERR_CNT | Error Control | EALLOW | Go |
| 12h | ERR_THRESHOLD | Error Threshold | EALLOW | Go |
| 14h | ERR_INTFLG | Error Interrupt Flag | EALLOW | Go |
| 16h | ERR_INTCLR | Error Interrupt Flag Clear | EALLOW | Go |
| 18h | FDATAH_TEST | Data High Test | EALLOW | Go |
| 1Ah | FDATAL_TEST | Data Low Test | EALLOW | Go |
| 1Ch | FADDR_TEST | ECC Test Address | EALLOW | Go |
| 1Eh | FECC_TEST | ECC Test Address | EALLOW | Go |
| 20h | FECC_CTRL | ECC Control | EALLOW | Go |
| 22h | FOUTH_TEST | Test Data Out High | EALLOW | Go |
| 24h | FOUTL_TEST | Test Data Out Low | EALLOW | Go |
| 26h | FECC_STATUS | ECC Status | EALLOW | Go |

Complex bit access types are encoded to fit into small table cells. Table 3-322 shows the codes that are used for access types in this section.

**Table 3-322. FLASH_ECC_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| R-0 | R<br>-0 | Read<br>Returns 0s |
| Write Type | | |
| W | W | Write |
| W1S | W<br>1S | Write<br>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. |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-322. FLASH_ECC_REGS Access Type Codes (continued)**

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



Copyright © 2024 Texas Instruments Incorporated

### 3.17.22.1 ECC_ENABLE Register (Offset = 0h) [Reset = 0000000Ah]

ECC_ENABLE is shown in Figure 3-286 and described in Table 3-323.

Return to the Summary Table.

ECC Enable

**Figure 3-286. ECC_ENABLE Register**

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

**Table 3-323. 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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.22.2 SINGLE_ERR_ADDR_LOW Register (Offset = 2h) [Reset = 00000000h]

SINGLE_ERR_ADDR_LOW is shown in Figure 3-287 and described in Table 3-324.

Return to the Summary Table.

Single Error Address Low

**Figure 3-287. SINGLE_ERR_ADDR_LOW Register**

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

**Table 3-324. SINGLE_ERR_ADDR_LOW Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | ERR_ADDR_L | R/W | 0h | 64-bit aligned address at which a single bit error occurred in the lower 64-bits of a 128-bit aligned memory.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.22.3 SINGLE_ERR_ADDR_HIGH Register (Offset = 4h) [Reset = 00000000h]

SINGLE_ERR_ADDR_HIGH is shown in Figure 3-288 and described in Table 3-325.

Return to the Summary Table.

Single Error Address High

**Figure 3-288. SINGLE_ERR_ADDR_HIGH Register**

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

**Table 3-325. SINGLE_ERR_ADDR_HIGH Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | ERR_ADDR_H | R/W | 0h | 64-bit aligned address at which a single bit error occurred in the upper 64-bits of a 128-bit aligned memory.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.22.4 UNC_ERR_ADDR_LOW Register (Offset = 6h) [Reset = 00000000h]

UNC_ERR_ADDR_LOW is shown in Figure 3-289 and described in Table 3-326.

Return to the Summary Table.

Uncorrectable Error Address Low

**Figure 3-289. UNC_ERR_ADDR_LOW Register**

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

**Table 3-326. UNC_ERR_ADDR_LOW Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | UNC_ERR_ADDR_L | R/W | 0h | 64-bit aligned address at which an uncorrectable error occurred in the lower 64-bits of a 128-bit aligned memory.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.22.5 UNC_ERR_ADDR_HIGH Register (Offset = 8h) [Reset = 00000000h]

UNC_ERR_ADDR_HIGH is shown in Figure 3-290 and described in Table 3-327.

Return to the Summary Table.

Uncorrectable Error Address High

**Figure 3-290. UNC_ERR_ADDR_HIGH Register**

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

**Table 3-327. UNC_ERR_ADDR_HIGH Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | UNC_ERR_ADDR_H | R/W | 0h | 64-bit aligned address at which an uncorrectable error occurred in the upper 64-bits of a 128-bit aligned memory.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.22.6 ERR_STATUS Register (Offset = Ah) [Reset = 00000000h]

ERR_STATUS is shown in Figure 3-291 and described in Table 3-328.

Return to the Summary Table.

Error Status

**Figure 3-291. ERR_STATUS Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | UNC_ERR_H | FAIL_1_H | FAIL_0_H |
| R-0h | | | | | R-0h | R-0h | R-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | UNC_ERR_L | FAIL_1_L | FAIL_0_L |
| R-0h | | | | | R-0h | R-0h | R-0h |

**Table 3-328. ERR_STATUS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-19 | RESERVED | R | 0h | Reserved |
| 18 | UNC_ERR_H | R | 0h | Uncorrectable error. A value of 1 indicates that an un-correctable error occurred in upper 64bits of a 128-bit aligned address. Cleared by writing a 1 to UNC_ERR_H_CLR bit of ERR_STATUS_CLR register.<br>Reset type: SYSRSn |
| 17 | FAIL_1_H | R | 0h | Fail on 1.<br>0 Fail on 1 single bit error did not occur in upper 64bits of a 128-bit aligned address.<br>1 A value of 1 would indicate that a single bit error occurred in upper 64bits of a 128-bit aligned address and the corrected value was 1. Cleared by writing a 1 to FAIL_1_H_CLR bit of ERR_STATUS_CLR register.<br>Note: This bit is updated on every flash access which results in a single bit error, So, in case of multiple single bit error, the status would correspond to the last error which occured.<br>Reset type: SYSRSn |
| 16 | FAIL_0_H | R | 0h | Fail on 0.<br>0 Fail on 0 single bit error did not occur in upper 64bits of a 128-bit aligned address.<br>1 A value of 1 would indicate that a single bit error occurred in upper 64bits of a 128-bit aligned address and the corrected value was 0. Cleared by writing a 1 to FAIL_0_H_CLR bit of ERR_STATUS_CLR register.<br>Note: This bit is updated on every flash access which results in a single bit error, So, in case of multiple single bit error, the status would correspond to the last error which occurred.<br>Reset type: SYSRSn |
| 15-3 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-328. ERR_STATUS Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 2 | UNC_ERR_L | R | 0h | Uncorrectable error. A value of 1 indicates that an un-correctable error occurred in lower 64bits of a 128-bit aligned address. Cleared by writing a 1 to UNC_ERR_L_CLR bit of ERR_STATUS_CLR register.<br>Reset type: SYSRSn |
| 1 | FAIL_1_L | R | 0h | Fail on 1.<br>0 Fail on 1 single bit error did not occur in lower 64bits of a 128-bit aligned address.<br>1 A value of 1 would indicate that a single bit error occurred in lower 64bits of a 128-bit aligned address and the corrected value was 1. Cleared by writing a 1 to FAIL_1_L_CLR bit of ERR_STATUS_CLR register.<br>Note: This bit is updated on every flash access which results in a single bit error, So, in case of multiple single bit error, the status would correspond to the last error which occured.<br>Reset type: SYSRSn |
| 0 | FAIL_0_L | R | 0h | Fail on 0.<br>0 Fail on 0 single bit error did not occur in lower 64bits of a 128-bit aligned address.<br>1 Would indicate that a single bit error occurred in lower 64bits of a 128-bit aligned address and the corrected value was 0. Cleared by writing a 1 to FAIL_0_L_CLR bit of ERR_STATUS_CLR register.<br>Note: This bit is updated on every flash access which results in a single bit error, So, in case of multiple single bit error, the status would correspond to the last error which occured.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.22.7 ERR_POS Register (Offset = Ch) [Reset = 00000000h]

ERR_POS is shown in Figure 3-292 and described in Table 3-329.

Return to the Summary Table.

Error Position

**Figure 3-292. ERR_POS Register**

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

**Table 3-329. ERR_POS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-25 | RESERVED | R | 0h | Reserved |
| 24 | ERR_TYPE_H | R | 0h | Error type<br>0 Indicates that a single bit error occured in upper 64 data bits of a 128-bit aligned address.<br>1 Indicates that a single bit error occured in ECC check bits of upper 64bits of a 128-bit aligned address.<br>Reset type: SYSRSn |
| 23-22 | RESERVED | R | 0h | Reserved |
| 21-16 | ERR_POS_H | R | 0h | Error position. Bit position of the single bit error in upper 64bits of a 128-bit aligned address. The position is interpreted depending on whether the ERR_TYPE bit indicates a check bit or a data bit. If ERR_TYPE indicates a check bit error, the error position could range from 0 to 7, else it could range from 0 to 63.<br>Reset type: SYSRSn |
| 15-9 | RESERVED | R | 0h | Reserved |
| 8 | ERR_TYPE_L | R | 0h | Error type<br>0 Indicates that a single bit error occured in lower 64 data bits of a 128-bit aligned address.<br>1 Indicates that a single bit error occured in ECC check bits of lower 64bits of a 128-bit aligned address.<br>Reset type: SYSRSn |
| 7-6 | RESERVED | R | 0h | Reserved |
| 5-0 | ERR_POS_L | R | 0h | Error position. Bit position of the single bit error in lower 64bits of a 128-bit aligned address. The position is interpreted depending on whether the ERR_TYPE bit indicates a check bit or a data bit. If ERR_TYPE indicates a check bit error, the error position could range from 0 to 7, else it could range from 0 to 63.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.22.8 ERR_STATUS_CLR Register (Offset = Eh) [Reset = 00000000h]

ERR_STATUS_CLR is shown in Figure 3-293 and described in Table 3-330.

Return to the Summary Table.

Error Status Clear

**Figure 3-293. ERR_STATUS_CLR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | UNC_ERR_H_CLR | FAIL_1_H_CLR | FAIL_0_H_CLR |
| R-0h | | | | | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | UNC_ERR_L_CLR | FAIL_1_L_CLR | FAIL_0_L_CLR |
| R-0h | | | | | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h |

**Table 3-330. ERR_STATUS_CLR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-19 | RESERVED | R | 0h | Reserved |
| 18 | UNC_ERR_H_CLR | R-0/W1S | 0h | Uncorrectable error clear. Writing a 1 to this bit will clear the UNC_ERR_H bit of ERR_STATUS register. Writes of 0 have no effect. Read returns 0.<br>Reset type: SYSRSn |
| 17 | FAIL_1_H_CLR | R-0/W1S | 0h | Fail on 1 clear. Writing a 1 to this bit will clear the FAIL_1_H bit of ERR_STATUS register. Writes of 0 have no effect. Read returns 0.<br>Reset type: SYSRSn |
| 16 | FAIL_0_H_CLR | R-0/W1S | 0h | Fail on 0 clear. Writing a 1 to this bit will clear the FAIL_0_H bit of ERR_STATUS register. Writes of 0 have no effect. Read returns 0.<br>Reset type: SYSRSn |
| 15-3 | RESERVED | R | 0h | Reserved |
| 2 | UNC_ERR_L_CLR | R-0/W1S | 0h | Uncorrectable error clear. Writing a 1 to this bit will clear the UNC_ERR_L bit of ERR_STATUS register. Writes of 0 have no effect. Read returns 0.<br>Reset type: SYSRSn |
| 1 | FAIL_1_L_CLR | R-0/W1S | 0h | Fail on 1 clear. Writing a 1 to this bit will clear the FAIL_1_L bit of ERR_STATUS register. Writes of 0 have no effect. Read returns 0.<br>Reset type: SYSRSn |
| 0 | FAIL_0_L_CLR | R-0/W1S | 0h | Fail on 0 clear. Writing a 1 to this bit will clear the FAIL_0_L bit of ERR_STATUS register. Writes of 0 have no effect. Read returns 0.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.22.9 ERR_CNT Register (Offset = 10h) [Reset = 00000000h]

ERR_CNT is shown in Figure 3-294 and described in Table 3-331.

Return to the Summary Table.

Error Control

**Figure 3-294. ERR_CNT Register**

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

**Table 3-331. ERR_CNT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-0 | ERR_CNT | R/W | 0h | Single bit error count. This counter increments with every single bit ECC error occurrence. Upon reaching the threshold value counter stops counting on single bit errors. ERR_CNT can be cleared (irrespective of whether threshold is met or not) using 'Single Err Int Clear' bit. This is applicable for ECC logic test mode and normal operational mode.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.22.10 ERR_THRESHOLD Register (Offset = 12h) [Reset = 00000000h]

ERR_THRESHOLD is shown in Figure 3-295 and described in Table 3-332.

Return to the Summary Table.

Error Threshold

**Figure 3-295. ERR_THRESHOLD Register**

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

**Table 3-332. ERR_THRESHOLD Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-0 | ERR_THRESHOLD | R/W | 0h | Single bit error threshold. Sets the threshold for single bit errors. When the ERR_CNT value equals the THRESHOLD value and a single bit error occurs, SINGLE_ERR_INT flag is set, and an interrupt is fired.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.22.11 ERR_INTFLG Register (Offset = 14h) [Reset = 00000000h]

ERR_INTFLG is shown in Figure 3-296 and described in Table 3-333.

Return to the Summary Table.

Error Interrupt Flag

**Figure 3-296. ERR_INTFLG Register**

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

**Table 3-333. ERR_INTFLG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-2 | RESERVED | R | 0h | Reserved |
| 1 | UNC_ERR_INTFLG | R | 0h | Uncorrectable bit error interrupt flag. When a Un-correctable error occurs, this bit is set and the UNC_ERR_INT interrupt is fired. When UNC_ERR_INTCLR bit of ERR_INTCLR register is written a value of 1 this bit is cleared.<br>Reset type: SYSRSn |
| 0 | SINGLE_ERR_INTFLG | R | 0h | Single bit error interrupt flag. When the ERR_CNT value equals the ERR_THRESHOLD value and a single bit error occurs then SINGLE_ERR_INT flag is set and SINGLE_ERR_INT interrupt is fired. When SINGLE_ERR_INTCLR bit of ERR_INTCLR register is written a value of 1 this bit is cleared.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.22.12 ERR_INTCLR Register (Offset = 16h) [Reset = 00000000h]

ERR_INTCLR is shown in Figure 3-297 and described in Table 3-334.

Return to the Summary Table.

Error Interrupt Flag Clear

**Figure 3-297. ERR_INTCLR Register**

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

**Table 3-334. ERR_INTCLR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-2 | RESERVED | R | 0h | Reserved |
| 1 | UNC_ERR_INTCLR | R-0/W1S | 0h | Uncorrectable bit error interrupt flag clear. Writing a 1 to this bit will clear UNC_ERR_INT_FLG. Writes of 0 have no effect.<br>Reset type: SYSRSn |
| 0 | SINGLE_ERR_INTCLR | R-0/W1S | 0h | Single bit error interrupt flag clear. Writing a 1 to this bit will clear SINGLE_ERR_INT_FLG. Writes of 0 have no effect.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.22.13 FDATAH_TEST Register (Offset = 18h) [Reset = 00000000h]

FDATAH_TEST is shown in Figure 3-298 and described in Table 3-335.

Return to the Summary Table.

Data High Test

**Figure 3-298. FDATAH_TEST Register**

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

**Table 3-335. FDATAH_TEST Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | FDATAH | R/W | 0h | High double word of selected 64-bit data. User-configurable bits 63:32 of the selected data block in ECC test mode.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.22.14 FDATAL_TEST Register (Offset = 1Ah) [Reset = 00000000h]

FDATAL_TEST is shown in Figure 3-299 and described in Table 3-336.

Return to the Summary Table.

Data Low Test

**Figure 3-299. FDATAL_TEST Register**

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

**Table 3-336. FDATAL_TEST Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | FDATAL | R/W | 0h | Low double word of selected 64-bit data. User-configurable bits 31:0 of the selected data block in ECC test mode.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.22.15 FADDR_TEST Register (Offset = 1Ch) [Reset = 00000000h]

FADDR_TEST is shown in Figure 3-300 and described in Table 3-337.

Return to the Summary Table.

ECC Test Address

**Figure 3-300. FADDR_TEST Register**

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

**Table 3-337. FADDR_TEST Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-22 | RESERVED | R | 0h | Reserved |
| 21-16 | ADDRH | R/W | 0h | Address for selected 64-bit data. User-configurable address bits of the selected data in ECC test mode. Left-shift the address by one bit (to provide byte address) and ignore the three least significant bits of the address and write the bits 21:16 in remaining address bits in this field.<br>Reset type: SYSRSn |
| 15-3 | ADDRL | R/W | 0h | Address for selected 64-bit data. User-configurable address bits of the selected data in ECC test mode. Left-shift the address by one bit (to provide byte address) and ignore the three least significant bits of the address and write the bits 15:3 in remaining address bits in this field.<br>Reset type: SYSRSn |
| 2-0 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.22.16 FECC_TEST Register (Offset = 1Eh) [Reset = 00000000h]

FECC_TEST is shown in Figure 3-301 and described in Table 3-338.

Return to the Summary Table.

ECC Test Address

**Figure 3-301. FECC_TEST Register**

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

**Table 3-338. FECC_TEST Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-8 | RESERVED | R | 0h | Reserved |
| 7-0 | ECC | R/W | 0h | 8-bit ECC for selected 64-bit data. User-configurable ECC bits of the selected 64-bit data block in ECC test mode.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.22.17 FECC_CTRL Register (Offset = 20h) [Reset = 00000000h]

FECC_CTRL is shown in Figure 3-302 and described in Table 3-339.

Return to the Summary Table.

ECC Control

**Figure 3-302. FECC_CTRL Register**

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

**Table 3-339. FECC_CTRL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-3 | RESERVED | R | 0h | Reserved |
| 2 | DO_ECC_CALC | R-0/W1S | 0h | Enable ECC calculation. ECC logic will calculate ECC in one cycle for the data and address written in ECC test registers when ECC test logic is enabled by setting ECC_TEST_EN.<br>Reset type: SYSRSn |
| 1 | ECC_SELECT | R/W | 0h | ECC block select.<br>0 Selects the ECC block on bits [63:0] of bank data.<br>1 Selects the ECC block on bits [127:64] of bank data.<br>Reset type: SYSRSn |
| 0 | ECC_TEST_EN | R/W | 0h | ECC test mode enable.<br>0 ECC test mode disabled<br>1 ECC test mode enabled<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.22.18 FOUTH_TEST Register (Offset = 22h) [Reset = 00000000h]

FOUTH_TEST is shown in Figure 3-303 and described in Table 3-340.

Return to the Summary Table.

Test Data Out High

**Figure 3-303. FOUTH_TEST Register**

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

**Table 3-340. FOUTH_TEST Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | DATAOUTH | R | 0h | High double word test data out. Holds bits 63:32 of the data out of the selected ECC block.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.22.19 FOUTL_TEST Register (Offset = 24h) [Reset = 00000000h]

FOUTL_TEST is shown in Figure 3-304 and described in Table 3-341.

Return to the Summary Table.

Test Data Out Low

**Figure 3-304. FOUTL_TEST Register**

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

**Table 3-341. FOUTL_TEST Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | DATAOUTL | R | 0h | Low double word test data out. Holds bits 31:0 of the data out of the selected ECC block.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.22.20 FECC_STATUS Register (Offset = 26h) [Reset = 00000000h]

FECC_STATUS is shown in Figure 3-305 and described in Table 3-342.

Return to the Summary Table.

ECC Status

**Figure 3-305. FECC_STATUS Register**

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

**Table 3-342. FECC_STATUS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-9 | RESERVED | R | 0h | Reserved |
| 8 | ERR_TYPE | R | 0h | Test mode ECC single bit error indicator. When 1, indicates that the single bit error is in check bits. When 0, indicates that the single bit error is in data bits (If SINGLE_ERR field is also set).<br>Reset type: SYSRSn |
| 7-2 | DATA_ERR_POS | R | 0h | Test mode single bit error position. Holds the bit position where the single bit error occurred.<br>The position is interpreted depending on whether the CHK_ERR bit indicates a check bit or a data bit. If CHK_ERR indicates a check bit error, the error position could range from 0 to 7, or it could range from 0 to 63.<br>Reset type: SYSRSn |
| 1 | UNC_ERR | R | 0h | Test mode ECC double bit error. When 1 indicates that the ECC test resulted in an uncorrectable bit error.<br>Reset type: SYSRSn |
| 0 | SINGLE_ERR | R | 0h | Test mode ECC single bit error. When 1 indicates that the ECC test resulted in a single bit error.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.23 CPU_ID_REGS Registers

Table 3-343 lists the memory-mapped registers for the CPU_ID_REGS registers. All register offset addresses not listed in Table 3-343 should be considered as reserved locations and the register contents should not be modified.

**Table 3-343. CPU_ID_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | CPUID | Indicates CPU1 or CPU2 | | Go |

Complex bit access types are encoded to fit into small table cells. Table 3-344 shows the codes that are used for access types in this section.

**Table 3-344. CPU_ID_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 |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.23.1 CPUID Register (Offset = 0h) [Reset = 0000h]

CPUID is shown in Figure 3-306 and described in Table 3-345.

Return to the Summary Table.

This register can be used to identify on which CPU the code is executing.

**Figure 3-306. CPUID Register**

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

**Table 3-345. CPUID Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R | 0h | Reserved |
| 7-0 | CPUID | R | 0h | CPUID = 1 for CPU1, 2 for CPU2<br>Reset type: N/A |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.24 UID_REGS Registers

Table 3-346 lists the memory-mapped registers for the UID_REGS registers. All register offset addresses not listed in Table 3-346 should be considered as reserved locations and the register contents should not be modified.

**Table 3-346. UID_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | UID_PSRAND0 | UID Psuedo-random 192 bit number | | Go |
| 2h | UID_PSRAND1 | UID Psuedo-random 192 bit number | | Go |
| 4h | UID_PSRAND2 | UID Psuedo-random 192 bit number | | Go |
| 6h | UID_PSRAND3 | UID Psuedo-random 192 bit number | | Go |
| 8h | UID_PSRAND4 | UID Psuedo-random 192 bit number | | Go |
| Ah | UID_PSRAND5 | UID Psuedo-random 192 bit number | | Go |
| Ch | UID_UNIQUE | UID Unique 32 bit number | | Go |
| Eh | UID_CHECKSUM | UID Checksum | | Go |

Complex bit access types are encoded to fit into small table cells. Table 3-347 shows the codes that are used for access types in this section.

**Table 3-347. 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 |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.24.1 UID_PSRAND0 Register (Offset = 0h) [Reset = 00000000h]

UID_PSRAND0 is shown in Figure 3-307 and described in Table 3-348.

Return to the Summary Table.

UID Psuedo-random 192 bit number

**Figure 3-307. UID_PSRAND0 Register**

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

**Table 3-348. UID_PSRAND0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | RandomID | R | 0h | Psuedorandom portion of the UID<br>Reset type: N/A |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.24.2 UID_PSRAND1 Register (Offset = 2h) [Reset = 00000000h]

UID_PSRAND1 is shown in Figure 3-308 and described in Table 3-349.

Return to the Summary Table.

UID Psuedo-random 192 bit number

**Figure 3-308. 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 3-349. UID_PSRAND1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | RandomID | R | 0h | Psuedorandom portion of the UID<br>Reset type: N/A |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.24.3 UID_PSRAND2 Register (Offset = 4h) [Reset = 00000000h]

UID_PSRAND2 is shown in Figure 3-309 and described in Table 3-350.

Return to the Summary Table.

UID Psuedo-random 192 bit number

**Figure 3-309. 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 3-350. UID_PSRAND2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | RandomID | R | 0h | Psuedorandom portion of the UID<br>Reset type: N/A |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.24.4 UID_PSRAND3 Register (Offset = 6h) [Reset = 00000000h]

UID_PSRAND3 is shown in Figure 3-310 and described in Table 3-351.

Return to the Summary Table.

UID Psuedo-random 192 bit number

**Figure 3-310. 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 3-351. UID_PSRAND3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | RandomID | R | 0h | Psuedorandom portion of the UID<br>Reset type: N/A |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.24.5 UID_PSRAND4 Register (Offset = 8h) [Reset = 00000000h]

UID_PSRAND4 is shown in Figure 3-311 and described in Table 3-352.

Return to the Summary Table.

UID Psuedo-random 192 bit number

**Figure 3-311. 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 3-352. UID_PSRAND4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | RandomID | R | 0h | Psuedorandom portion of the UID<br>Reset type: N/A |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.24.6 UID_PSRAND5 Register (Offset = Ah) [Reset = 00000000h]

UID_PSRAND5 is shown in Figure 3-312 and described in Table 3-353.

Return to the Summary Table.

UID Psuedo-random 192 bit number

**Figure 3-312. UID_PSRAND5 Register**

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

**Table 3-353. UID_PSRAND5 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | RandomID | R | 0h | Psuedorandom portion of the UID<br>Reset type: N/A |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.24.7 UID_UNIQUE Register (Offset = Ch) [Reset = 00000000h]

UID_UNIQUE is shown in Figure 3-313 and described in Table 3-354.

Return to the Summary Table.

UID Unique 32 bit number

**Figure 3-313. UID_UNIQUE Register**

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

**Table 3-354. UID_UNIQUE 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.<br>Reset type: N/A |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.24.8 UID_CHECKSUM Register (Offset = Eh) [Reset = 00000000h]

UID_CHECKSUM is shown in Figure 3-314 and described in Table 3-355.

Return to the Summary Table.

Fletcher checksum of UID_PSRAND and UID_UNIQUE registers

**Figure 3-314. 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 3-355. UID_CHECKSUM Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | Checksum | R | 0h | Fletcher checksum of UID_PSRANDx and UID_UINIQUE<br>Reset type: N/A |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.25 DCSM_Z1_OTP Registers

Table 3-356 lists the memory-mapped registers for the DCSM_Z1_OTP registers. All register offset addresses not listed in Table 3-356 should be considered as reserved locations and the register contents should not be modified.

**Table 3-356. DCSM_Z1_OTP Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | Z1OTP_LINKPOINTER1 | Zone 1 Link Pointer1 in Z1 OTP | | Go |
| 4h | Z1OTP_LINKPOINTER2 | Zone 1 Link Pointer2 in Z1 OTP | | Go |
| 8h | Z1OTP_LINKPOINTER3 | Zone 1 Link Pointer3 in Z1 OTP | | Go |
| 10h | Z1OTP_PSWDLOCK | Secure Password Lock in Z1 OTP | | Go |
| 14h | Z1OTP_CRCLOCK | Secure CRC Lock in Z1 OTP | | Go |
| 18h | Z1OTP_JTAGLOCK | Secure JTAG Lock in Z1 OTP | | Go |
| 1Eh | Z1OTP_BOOTCTRL | Boot Mode in Z1 OTP | | Go |

Complex bit access types are encoded to fit into small table cells. Table 3-357 shows the codes that are used for access types in this section.

**Table 3-357. 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 |
| 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. |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.25.1 Z1OTP_LINKPOINTER1 Register (Offset = 0h) [Reset = FFFFFFFFh]

Z1OTP_LINKPOINTER1 is shown in Figure 3-315 and described in Table 3-358.

Return to the Summary Table.

Zone 1 Link Pointer1 in Z1 OTP

**Figure 3-315. 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-FFFFFFFFh | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 3-358. Z1OTP_LINKPOINTER1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | Z1OTP_LINKPOINTER1 | R | FFFFFFFFh | Zone1 Link Pointer 1 location in USER OTP.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.25.2 Z1OTP_LINKPOINTER2 Register (Offset = 4h) [Reset = FFFFFFFFh]

Z1OTP_LINKPOINTER2 is shown in Figure 3-316 and described in Table 3-359.

Return to the Summary Table.

Zone 1 Link Pointer2 in Z1 OTP

**Figure 3-316. 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-FFFFFFFFh | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 3-359. Z1OTP_LINKPOINTER2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | Z1OTP_LINKPOINTER2 | R | FFFFFFFFh | Zone1 Link Pointer 2 location in USER OTP.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.25.3 Z1OTP_LINKPOINTER3 Register (Offset = 8h) [Reset = FFFFFFFFh]

Z1OTP_LINKPOINTER3 is shown in Figure 3-317 and described in Table 3-360.

Return to the Summary Table.

Zone 1 Link Pointer3 in Z1 OTP

**Figure 3-317. 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-FFFFFFFFh | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 3-360. Z1OTP_LINKPOINTER3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | Z1OTP_LINKPOINTER3 | R | FFFFFFFFh | Zone1 Link Pointer 3 location in USER OTP.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.25.4 Z1OTP_PSWDLOCK Register (Offset = 10h) [Reset = FFFFFFFFh]

Z1OTP_PSWDLOCK is shown in Figure 3-318 and described in Table 3-361.

Return to the Summary Table.

Secure Password Lock in Z1 OTP

**Figure 3-318. 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-FFFFFFFFh | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 3-361. Z1OTP_PSWDLOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | Z1OTP_PSWDLOCK | R | FFFFFFFFh | Zone1 password lock location in USER OTP.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.25.5 Z1OTP_CRCLOCK Register (Offset = 14h) [Reset = FFFFFFFFh]

Z1OTP_CRCLOCK is shown in Figure 3-319 and described in Table 3-362.

Return to the Summary Table.

Secure CRC Lock in Z1 OTP

**Figure 3-319. Z1OTP_CRCLOCK Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Z1OTP_CRCLOCK | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R-FFFFFFFFh | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 3-362. Z1OTP_CRCLOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | Z1OTP_CRCLOCK | R | FFFFFFFFh | Zone1 CRC lock location in USER OTP.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.25.6 Z1OTP_JTAGLOCK Register (Offset = 18h) [Reset = FFFFFFFFh]

Z1OTP_JTAGLOCK is shown in Figure 3-320 and described in Table 3-363.

Return to the Summary Table.

Secure JTAG Lock in Z1 OTP

**Figure 3-320. Z1OTP_JTAGLOCK Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 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_JTAGLOCK | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R-FFFFFFFFh | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 3-363. Z1OTP_JTAGLOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | Z1OTP_JTAGLOCK | R | FFFFFFFFh | Zone1 JTAG lock location in USER OTP.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.25.7 Z1OTP_BOOTCTRL Register (Offset = 1Eh) [Reset = FFFFFFFFh]

Z1OTP_BOOTCTRL is shown in Figure 3-321 and described in Table 3-364.

Return to the Summary Table.

Boot Mode in Z1 OTP

**Figure 3-321. Z1OTP_BOOTCTRL Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Z1OTP_BOOTCTRL | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R-FFFFFFFFh | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 3-364. Z1OTP_BOOTCTRL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | Z1OTP_BOOTCTRL | R | FFFFFFFFh | Zone1 Boot control location in USER OTP.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.26 DCSM_Z2_OTP Registers

Table 3-365 lists the memory-mapped registers for the DCSM_Z2_OTP registers. All register offset addresses not listed in Table 3-365 should be considered as reserved locations and the register contents should not be modified.

**Table 3-365. DCSM_Z2_OTP Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | Z2OTP_LINKPOINTER1 | Zone 2 Link Pointer1 in Z2 OTP | | Go |
| 4h | Z2OTP_LINKPOINTER2 | Zone 2 Link Pointer2 in Z2 OTP | | Go |
| 8h | Z2OTP_LINKPOINTER3 | Zone 2 Link Pointer3 in Z2 OTP | | Go |
| 10h | Z2OTP_PSWDLOCK | Secure Password Lock in Z2 OTP | | Go |
| 14h | Z2OTP_CRCLOCK | Secure CRC Lock in Z2 OTP | | Go |
| 18h | Z2OTP_JTAGLOCK | Secure JTAG Lock in Z2 OTP | | Go |
| 1Eh | Z2OTP_BOOTCTRL | Boot Mode in Z2 OTP | | Go |

Complex bit access types are encoded to fit into small table cells. Table 3-366 shows the codes that are used for access types in this section.

**Table 3-366. 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 |
| 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. |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.26.1 Z2OTP_LINKPOINTER1 Register (Offset = 0h) [Reset = FFFFFFFFh]

Z2OTP_LINKPOINTER1 is shown in Figure 3-322 and described in Table 3-367.

Return to the Summary Table.

Zone 2 Link Pointer1 in Z2 OTP

**Figure 3-322. 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-FFFFFFFFh | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 3-367. Z2OTP_LINKPOINTER1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | Z2OTP_LINKPOINTER1 | R | FFFFFFFFh | Zone2 Link Pointer 1 location in USER OTP.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.26.2 Z2OTP_LINKPOINTER2 Register (Offset = 4h) [Reset = FFFFFFFFh]

Z2OTP_LINKPOINTER2 is shown in Figure 3-323 and described in Table 3-368.

Return to the Summary Table.

Zone 2 Link Pointer2 in Z2 OTP

**Figure 3-323. 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-FFFFFFFFh | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 3-368. Z2OTP_LINKPOINTER2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | Z2OTP_LINKPOINTER2 | R | FFFFFFFFh | Zone2 Link Pointer 2 location in USER OTP.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.26.3 Z2OTP_LINKPOINTER3 Register (Offset = 8h) [Reset = FFFFFFFFh]

Z2OTP_LINKPOINTER3 is shown in Figure 3-324 and described in Table 3-369.

Return to the Summary Table.

Zone 2 Link Pointer3 in Z2 OTP

**Figure 3-324. 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-FFFFFFFFh | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 3-369. Z2OTP_LINKPOINTER3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | Z2OTP_LINKPOINTER3 | R | FFFFFFFFh | Zone2 Link Pointer 3 location in USER OTP.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.26.4 Z2OTP_PSWDLOCK Register (Offset = 10h) [Reset = FFFFFFFFh]

Z2OTP_PSWDLOCK is shown in Figure 3-325 and described in Table 3-370.

Return to the Summary Table.

Secure Password Lock in Z2 OTP

**Figure 3-325. 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-FFFFFFFFh | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 3-370. Z2OTP_PSWDLOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | Z2OTP_PSWDLOCK | R | FFFFFFFFh | Zone2 password lock location in USER OTP.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.26.5 Z2OTP_CRCLOCK Register (Offset = 14h) [Reset = FFFFFFFFh]

Z2OTP_CRCLOCK is shown in Figure 3-326 and described in Table 3-371.

Return to the Summary Table.

Secure CRC Lock in Z2 OTP

**Figure 3-326. Z2OTP_CRCLOCK Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Z2OTP_CRCLOCK | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R-FFFFFFFFh | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 3-371. Z2OTP_CRCLOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | Z2OTP_CRCLOCK | R | FFFFFFFFh | Zone2 CRC lock location in USER OTP.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.26.6 Z2OTP_JTAGLOCK Register (Offset = 18h) [Reset = FFFFFFFFh]

Z2OTP_JTAGLOCK is shown in Figure 3-327 and described in Table 3-372.

Return to the Summary Table.

Secure JTAG Lock in Z2 OTP

**Figure 3-327. Z2OTP_JTAGLOCK Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 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_JTAGLOCK | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R-FFFFFFFFh | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 3-372. Z2OTP_JTAGLOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | Z2OTP_JTAGLOCK | R | FFFFFFFFh | Zone2 JTAG lock location in USER OTP.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 3.17.26.7 Z2OTP_BOOTCTRL Register (Offset = 1Eh) [Reset = FFFFFFFFh]

Z2OTP_BOOTCTRL is shown in Figure 3-328 and described in Table 3-373.

Return to the Summary Table.

Boot Mode in Z2 OTP

**Figure 3-328. Z2OTP_BOOTCTRL Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Z2OTP_BOOTCTRL | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R-FFFFFFFFh | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 3-373. Z2OTP_BOOTCTRL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | Z2OTP_BOOTCTRL | R | FFFFFFFFh | Zone2 Boot control location in USER OTP.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

## 3.17.27 Register to Driverlib Function Mapping

### 3.17.27.1 CPUTIMER Registers to Driverlib Functions

**Table 3-374. CPUTIMER Registers to Driverlib Functions**

| File | Driverlib Function |
|---|---|
| **TIM** | |
| cputimer.h | CPUTimer_getTimerCount |
| sysctl.c | SysCtl_setClock |
| **PRD** | |
| cputimer.h | CPUTimer_setPeriod |
| sysctl.c | SysCtl_setClock |
| sysctl.c | SysCtl_setAuxClock |
| sysctl.c | SysCtl_selectXTAL |
| **TCR** | |
| 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 |
| sysctl.c | SysCtl_setClock |
| sysctl.c | SysCtl_setAuxClock |
| sysctl.c | SysCtl_selectXTAL |
| **TPR** | |
| cputimer.h | CPUTimer_setPreScaler |
| sysctl.c | SysCtl_setClock |
| sysctl.c | SysCtl_setAuxClock |
| sysctl.c | SysCtl_selectXTAL |
| **TPRH** | |
| cputimer.h | CPUTimer_setPreScaler |
| sysctl.c | SysCtl_setClock |
| sysctl.c | SysCtl_setAuxClock |
| sysctl.c | SysCtl_selectXTAL |

### 3.17.27.2 ASYSCTL Registers to Driverlib Functions

**Table 3-375. ASYSCTL Registers to Driverlib Functions**

| File | Driverlib Function |
|---|---|
| **INTOSC1TRIM** | |
| - | |
| **INTOSC2TRIM** | |
| - | |
| **TSNSCTL** | |
| asysctl.h | ASysCtl_enableTemperatureSensor |
| asysctl.h | ASysCtl_disableTemperatureSensor |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-375. ASYSCTL Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| **LOCK** | |
| asysctl.h | ASysCtl_lockTemperatureSensor |
| **ANAREFTRIMA** | |
| - | |
| **ANAREFTRIMB** | |
| - | |
| **ANAREFTRIMC** | |
| - | |
| **ANAREFTRIMD** | |
| - | |

### 3.17.27.3 PIE Registers to Driverlib Functions

**Table 3-376. PIE Registers to Driverlib Functions**

| File | Driverlib Function |
|---|---|
| **CTRL** | |
| interrupt.c | Interrupt_initModule |
| interrupt.c | Interrupt_defaultHandler |
| interrupt.h | Interrupt_enablePIE |
| interrupt.h | Interrupt_disablePIE |
| **ACK** | |
| interrupt.c | Interrupt_disable |
| interrupt.h | Interrupt_clearACKGroup |
| **IER1** | |
| interrupt.c | Interrupt_initModule |
| interrupt.c | Interrupt_enable |
| interrupt.c | Interrupt_disable |
| **IFR1** | |
| interrupt.c | Interrupt_initModule |
| **IER2** | |
| interrupt.c | Interrupt_initModule |
| **IFR2** | |
| interrupt.c | Interrupt_initModule |
| **IER3** | |
| interrupt.c | Interrupt_initModule |
| **IFR3** | |
| interrupt.c | Interrupt_initModule |
| **IER4** | |
| interrupt.c | Interrupt_initModule |
| **IFR4** | |
| interrupt.c | Interrupt_initModule |
| **IER5** | |
| interrupt.c | Interrupt_initModule |
| **IFR5** | |
| interrupt.c | Interrupt_initModule |
| **IER6** | |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-376. PIE Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| interrupt.c | Interrupt_initModule |
| **IFR6** | |
| interrupt.c | Interrupt_initModule |
| **IER7** | |
| interrupt.c | Interrupt_initModule |
| **IFR7** | |
| interrupt.c | Interrupt_initModule |
| **IER8** | |
| interrupt.c | Interrupt_initModule |
| **IFR8** | |
| interrupt.c | Interrupt_initModule |
| **IER9** | |
| interrupt.c | Interrupt_initModule |
| **IFR9** | |
| interrupt.c | Interrupt_initModule |
| **IER10** | |
| interrupt.c | Interrupt_initModule |
| **IFR10** | |
| interrupt.c | Interrupt_initModule |
| **IER11** | |
| interrupt.c | Interrupt_initModule |
| **IFR11** | |
| interrupt.c | Interrupt_initModule |
| **IER12** | |
| interrupt.c | Interrupt_initModule |
| **IFR12** | |
| interrupt.c | Interrupt_initModule |

### 3.17.27.4 SYSCTL Registers to Driverlib Functions

**Table 3-377. SYSCTL Registers to Driverlib Functions**

| File | Driverlib Function |
|---|---|
| **DEVCFGLOCK1** | |
| sysctl.h | SysCtl_lockCPUSelectRegs |
| **PARTIDL** | |
| sysctl.c | SysCtl_getDeviceParametric |
| **PARTIDH** | |
| sysctl.c | SysCtl_getDeviceParametric |
| **REVID** | |
| adc.h | ADC_getTemperatureC |
| adc.h | ADC_getTemperatureK |
| sysctl.h | SysCtl_getDeviceRevision |
| **DC0** | |
| - | |
| **DC1** | |
| - | |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-377. SYSCTL Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| **DC2** | |
| - | |
| **DC3** | |
| - | |
| **DC4** | |
| - | |
| **DC5** | |
| - | |
| **DC6** | |
| - | |
| **DC7** | |
| - | |
| **DC8** | |
| - | |
| **DC9** | |
| - | |
| **DC10** | |
| - | |
| **DC11** | |
| - | |
| **DC12** | |
| - | |
| **DC13** | |
| - | |
| **DC14** | |
| - | |
| **DC15** | |
| - | |
| **DC17** | |
| - | |
| **DC18** | |
| - | |
| **DC19** | |
| - | |
| **DC20** | |
| - | |
| **PERCNF1** | |
| sysctl.h | SysCtl_isPresentUSBPHY |
| **FUSEERR** | |
| sysctl.h | SysCtl_getEfuseError |
| **SOFTPRES0** | |
| sysctl.h | SysCtl_resetPeripheral |
| **SOFTPRES1** | |
| - | See SOFTPRES0 |
| **SOFTPRES2** | |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-377. SYSCTL Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| - | See SOFTPRES0 |
| **SOFTPRES3** | |
| - | See SOFTPRES0 |
| **SOFTPRES4** | |
| - | See SOFTPRES0 |
| **SOFTPRES6** | |
| - | See SOFTPRES0 |
| **SOFTPRES7** | |
| - | See SOFTPRES0 |
| **SOFTPRES8** | |
| - | See SOFTPRES0 |
| **SOFTPRES9** | |
| - | See SOFTPRES0 |
| **SOFTPRES11** | |
| - | See SOFTPRES0 |
| **SOFTPRES13** | |
| - | See SOFTPRES0 |
| **SOFTPRES14** | |
| - | See SOFTPRES0 |
| **SOFTPRES16** | |
| - | See SOFTPRES0 |
| **CPUSEL0** | |
| sysctl.h | SysCtl_selectCPUForPeripheralInstance |
| sysctl.h | SysCtl_selectCPUForPeripheral |
| **CPUSEL1** | |
| - | See CPUSEL0 |
| **CPUSEL2** | |
| - | See CPUSEL0 |
| **CPUSEL4** | |
| - | See CPUSEL0 |
| **CPUSEL5** | |
| - | See CPUSEL0 |
| **CPUSEL6** | |
| - | See CPUSEL0 |
| **CPUSEL7** | |
| - | See CPUSEL0 |
| **CPUSEL8** | |
| - | See CPUSEL0 |
| **CPUSEL9** | |
| - | See CPUSEL0 |
| **CPUSEL11** | |
| - | See CPUSEL0 |
| **CPUSEL12** | |
| - | See CPUSEL0 |
| **CPUSEL14** | |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-377. SYSCTL Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| - | See CPUSEL0 |
| **CPU2RESCTL** | |
| - | |
| **RSTSTAT** | |
| - | |
| **LPMSTAT** | |
| - | |
| **SYSDBGCTL** | |
| sysctl.c | SysCtl_setClock |
| **CLKSEM** | |
| - | |
| **CLKCFGLOCK1** | |
| - | |
| **CLKSRCCTL1** | |
| sysctl.c | SysCtl_getClock |
| sysctl.c | SysCtl_selectXTAL |
| sysctl.c | SysCtl_selectOscSource |
| sysctl.c | SysCtl_selectOscSourceAuxPLL |
| sysctl.h | SysCtl_turnOnOsc |
| sysctl.h | SysCtl_turnOffOsc |
| sysctl.h | SysCtl_enableWatchdogInHalt |
| sysctl.h | SysCtl_disableWatchdogInHalt |
| **CLKSRCCTL2** | |
| can.h | CAN_selectClockSource |
| sysctl.c | SysCtl_getAuxClock |
| sysctl.c | SysCtl_selectXTAL |
| sysctl.c | SysCtl_selectOscSourceAuxPLL |
| **CLKSRCCTL3** | |
| sysctl.h | SysCtl_selectClockOutSource |
| **SYSPLLCTL1** | |
| sysctl.c | SysCtl_getClock |
| sysctl.c | SysCtl_setClock |
| sysctl.h | SysCtl_enterHaltMode |
| sysctl.h | SysCtl_enterHibernateMode |
| **SYSPLLMULT** | |
| sysctl.c | SysCtl_getClock |
| sysctl.c | SysCtl_setClock |
| **SYSPLLSTS** | |
| sysctl.c | SysCtl_setClock |
| **AUXPLLCTL1** | |
| sysctl.c | SysCtl_getAuxClock |
| sysctl.c | SysCtl_setAuxClock |
| sysctl.c | SysCtl_selectXTAL |
| **AUXPLLMULT** | |
| sysctl.c | SysCtl_getAuxClock |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-377. SYSCTL Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| sysctl.c | SysCtl_setAuxClock |
| **AUXPLLSTS** | |
| sysctl.c | SysCtl_setAuxClock |
| **SYSCLKDIVSEL** | |
| sysctl.c | SysCtl_getClock |
| sysctl.c | SysCtl_setClock |
| sysctl.h | SysCtl_setPLLSysClk |
| **AUXCLKDIVSEL** | |
| sysctl.c | SysCtl_getAuxClock |
| sysctl.c | SysCtl_setAuxClock |
| sysctl.c | SysCtl_selectXTAL |
| sysctl.h | SysCtl_setAuxPLLClk |
| **PERCLKDIVSEL** | |
| sysctl.h | SysCtl_setEPWMClockDivider |
| sysctl.h | SysCtl_setEMIF1ClockDivider |
| sysctl.h | SysCtl_setEMIF2ClockDivider |
| **XCLKOUTDIVSEL** | |
| sysctl.h | SysCtl_setXClk |
| **LOSPCP** | |
| sysctl.c | SysCtl_getLowSpeedClock |
| sysctl.h | SysCtl_setLowSpeedClock |
| **MCDCR** | |
| sysctl.h | SysCtl_enableMCD |
| sysctl.h | SysCtl_disableMCD |
| sysctl.h | SysCtl_isMCDClockFailureDetected |
| sysctl.h | SysCtl_resetMCD |
| sysctl.h | SysCtl_connectMCDClockSource |
| sysctl.h | SysCtl_disconnectMCDClockSource |
| **X1CNT** | |
| sysctl.h | SysCtl_getExternalOscCounterValue |
| **CPUSYSLOCK1** | |
| - | |
| **HIBBOOTMODE** | |
| - | |
| **IORESTOREADDR** | |
| - | |
| **PIEVERRADDR** | |
| sysctl.h | SysCtl_getPIEVErrAddr |
| **PCLKCR0** | |
| sysctl.h | SysCtl_enablePeripheral |
| sysctl.h | SysCtl_disablePeripheral |
| **PCLKCR1** | |
| - | See PCLKCR0 |
| **PCLKCR2** | |
| - | See PCLKCR0 |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-377. SYSCTL Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| **PCLKCR3** | |
| - | See PCLKCR0 |
| **PCLKCR4** | |
| - | See PCLKCR0 |
| **PCLKCR6** | |
| - | See PCLKCR0 |
| **PCLKCR7** | |
| - | See PCLKCR0 |
| **PCLKCR8** | |
| - | See PCLKCR0 |
| **PCLKCR9** | |
| - | See PCLKCR0 |
| **PCLKCR10** | |
| - | See PCLKCR0 |
| **PCLKCR11** | |
| - | See PCLKCR0 |
| **PCLKCR12** | |
| - | See PCLKCR0 |
| **PCLKCR13** | |
| - | See PCLKCR0 |
| **PCLKCR14** | |
| - | See PCLKCR0 |
| **PCLKCR16** | |
| - | See PCLKCR0 |
| **SECMSEL** | |
| sysctl.h | SysCtl_selectSecController |
| **LPMCR** | |
| sysctl.h | SysCtl_enterIdleMode |
| sysctl.h | SysCtl_enterStandbyMode |
| sysctl.h | SysCtl_enterHaltMode |
| sysctl.h | SysCtl_enterHibernateMode |
| sysctl.h | SysCtl_setStandbyQualificationPeriod |
| sysctl.h | SysCtl_enableWatchdogStandbyWakeup |
| sysctl.h | SysCtl_disableWatchdogStandbyWakeup |
| **GPIOLPMSEL0** | |
| sysctl.h | SysCtl_enableLPMWakeupPin |
| sysctl.h | SysCtl_disableLPMWakeupPin |
| **GPIOLPMSEL1** | |
| sysctl.h | SysCtl_enableLPMWakeupPin |
| sysctl.h | SysCtl_disableLPMWakeupPin |
| **TMR2CLKCTL** | |
| cputimer.h | CPUTimer_selectClockSource |
| sysctl.c | SysCtl_setClock |
| sysctl.c | SysCtl_setAuxClock |
| sysctl.c | SysCtl_selectXTAL |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-377. SYSCTL Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| sysctl.h | SysCtl_setCputimer2Clk |
| **RESC** | |
| sysctl.h | SysCtl_getResetCause |
| sysctl.h | SysCtl_clearResetCause |
| sysctl.h | SysCtl_getWatchdogResetStatus |
| sysctl.h | SysCtl_clearWatchdogResetStatus |
| **SCSR** | |
| sysctl.c | SysCtl_setClock |
| sysctl.h | SysCtl_setWatchdogMode |
| sysctl.h | SysCtl_isWatchdogInterruptActive |
| sysctl.h | SysCtl_clearWatchdogOverride |
| **WDCNTR** | |
| sysctl.h | SysCtl_getWatchdogCounterValue |
| **WDKEY** | |
| sysctl.h | SysCtl_serviceWatchdog |
| sysctl.h | SysCtl_enableWatchdogReset |
| sysctl.h | SysCtl_resetWatchdog |
| **WDCR** | |
| sysctl.c | SysCtl_setClock |
| sysctl.h | SysCtl_resetDevice |
| sysctl.h | SysCtl_disableWatchdog |
| sysctl.h | SysCtl_enableWatchdog |
| sysctl.h | SysCtl_isWatchdogEnabled |
| sysctl.h | SysCtl_setWatchdogPrescaler |
| **WDWCR** | |
| sysctl.c | SysCtl_setClock |
| sysctl.h | SysCtl_setWatchdogWindowValue |
| **CLA1TASKSRCSELLOCK** | |
| - | |
| **DMACHSRCSELLOCK** | |
| - | |
| **CLA1TASKSRCSEL1** | |
| cla.c | CLA_setTriggerSource |
| **CLA1TASKSRCSEL2** | |
| cla.c | CLA_setTriggerSource |
| **DMACHSRCSEL1** | |
| dma.c | DMA_configMode |
| **DMACHSRCSEL2** | |
| dma.c | DMA_configMode |
| **SYNCSELECT** | |
| sysctl.h | SysCtl_setSyncInputConfig |
| sysctl.h | SysCtl_setSyncOutputConfig |
| **ADCSOCOUTSELECT** | |
| sysctl.h | SysCtl_enableExtADCSOCSource |
| sysctl.h | SysCtl_disableExtADCSOCSource |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-377. SYSCTL Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| **SYNCSOCLOCK** | |
| sysctl.h | SysCtl_lockExtADCSOCSelect |
| sysctl.h | SysCtl_lockSyncSelect |

### 3.17.27.5 NMI Registers to Driverlib Functions

**Table 3-378. NMI Registers to Driverlib Functions**

| File | Driverlib Function |
|---|---|
| **CFG** | |
| sysctl.h | SysCtl_enableNMIGlobalInterrupt |
| **FLG** | |
| sysctl.h | SysCtl_getNMIStatus |
| sysctl.h | SysCtl_getNMIFlagStatus |
| sysctl.h | SysCtl_isNMIFlagSet |
| sysctl.h | SysCtl_clearNMIStatus |
| sysctl.h | SysCtl_clearAllNMIFlags |
| sysctl.h | SysCtl_forceNMIFlags |
| **FLGCLR** | |
| sysctl.h | SysCtl_clearNMIStatus |
| sysctl.h | SysCtl_clearAllNMIFlags |
| **FLGFRC** | |
| sysctl.h | SysCtl_forceNMIFlags |
| **WDCNT** | |
| sysctl.h | SysCtl_getNMIWatchdogCounter |
| **WDPRD** | |
| sysctl.h | SysCtl_setNMIWatchdogPeriod |
| sysctl.h | SysCtl_getNMIWatchdogPeriod |
| **SHDFLG** | |
| sysctl.h | SysCtl_getNMIShadowFlagStatus |
| sysctl.h | SysCtl_isNMIShadowFlagSet |

### 3.17.27.6 XINT Registers to Driverlib Functions

**Table 3-379. XINT Registers to Driverlib Functions**

| File | Driverlib Function |
|---|---|
| **1CR** | |
| gpio.c | GPIO_setInterruptPin |
| gpio.h | GPIO_setInterruptType |
| gpio.h | GPIO_getInterruptType |
| gpio.h | GPIO_enableInterrupt |
| gpio.h | GPIO_disableInterrupt |
| gpio.h | GPIO_getInterruptCounter |
| **2CR** | |
| - | See 1CR |
| **3CR** | |
| - | See 1CR |
| **4CR** | |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-379. XINT Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| - | See 1CR |
| **5CR** | |
| - | See 1CR |
| **1CTR** | |
| gpio.h | GPIO_getInterruptCounter |
| **2CTR** | |
| - | |
| **3CTR** | |
| - | |

### 3.17.27.7 DCSM Registers to Driverlib Functions

**Table 3-380. DCSM Registers to Driverlib Functions**

| File | Driverlib Function |
|---|---|
| **Z1OTP_LINKPOINTER1** | |
| - | |
| **Z1OTP_LINKPOINTER2** | |
| - | |
| **Z1OTP_LINKPOINTER3** | |
| - | |
| **Z1OTP_PSWDLOCK** | |
| - | |
| **Z1OTP_CRCLOCK** | |
| - | |
| **Z1OTP_BOOTCTRL** | |
| - | |
| **Z2OTP_LINKPOINTER1** | |
| - | |
| **Z2OTP_LINKPOINTER2** | |
| - | |
| **Z2OTP_LINKPOINTER3** | |
| - | |
| **Z2OTP_PSWDLOCK** | |
| - | |
| **Z2OTP_CRCLOCK** | |
| - | |
| **Z2OTP_BOOTCTRL** | |
| - | |
| **Z1_LINKPOINTER** | |
| dcsm.c | DCSM_unlockZone1CSM |
| dcsm.h | DCSM_getZone1LinkPointerError |
| **Z1_OTPSECLOCK** | |
| - | |
| **Z1_BOOTCTRL** | |
| - | |
| **Z1_LINKPOINTERERR** | |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-380. DCSM Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| dcsm.h | DCSM_getZone1LinkPointerError |
| **Z1_CSMKEY0** | |
| dcsm.c | DCSM_unlockZone1CSM |
| **Z1_CSMKEY1** | |
| dcsm.c | DCSM_unlockZone1CSM |
| **Z1_CSMKEY2** | |
| dcsm.c | DCSM_unlockZone1CSM |
| **Z1_CSMKEY3** | |
| dcsm.c | DCSM_unlockZone1CSM |
| **Z1_CR** | |
| dcsm.h | DCSM_secureZone1 |
| dcsm.h | DCSM_getZone1CSMSecurityStatus |
| dcsm.h | DCSM_getZone1ControlStatus |
| **Z1_GRABSECTR** | |
| - | |
| **Z1_GRABRAMR** | |
| - | |
| **Z1_EXEONLYSECTR** | |
| dcsm.c | DCSM_getZone1FlashEXEStatus |
| **Z1_EXEONLYRAMR** | |
| dcsm.c | DCSM_getZone1RAMEXEStatus |
| **Z2_LINKPOINTER** | |
| dcsm.c | DCSM_unlockZone2CSM |
| dcsm.h | DCSM_getZone2LinkPointerError |
| **Z2_OTPSECLOCK** | |
| - | |
| **Z2_BOOTCTRL** | |
| - | |
| **Z2_LINKPOINTERERR** | |
| dcsm.h | DCSM_getZone2LinkPointerError |
| **Z2_CSMKEY0** | |
| dcsm.c | DCSM_unlockZone2CSM |
| **Z2_CSMKEY1** | |
| dcsm.c | DCSM_unlockZone2CSM |
| **Z2_CSMKEY2** | |
| dcsm.c | DCSM_unlockZone2CSM |
| **Z2_CSMKEY3** | |
| dcsm.c | DCSM_unlockZone2CSM |
| **Z2_CR** | |
| dcsm.h | DCSM_secureZone2 |
| dcsm.h | DCSM_getZone2CSMSecurityStatus |
| dcsm.h | DCSM_getZone2ControlStatus |
| **Z2_GRABSECTR** | |
| - | |
| **Z2_GRABRAMR** | |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-380. DCSM Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| - | |
| **Z2_EXEONLYSECTR** | |
| dcsm.c | DCSM_getZone2FlashEXEStatus |
| **Z2_EXEONLYRAMR** | |
| dcsm.c | DCSM_getZone2RAMEXEStatus |
| **FLSEM** | |
| dcsm.c | DCSM_claimZoneSemaphore |
| dcsm.c | DCSM_releaseZoneSemaphore |
| **SECTSTAT** | |
| dcsm.h | DCSM_getFlashSectorZone |
| **RAMSTAT** | |
| dcsm.h | DCSM_getRAMZone |

### 3.17.27.8 MEMCFG Registers to Driverlib Functions

**Table 3-381. MEMCFG Registers to Driverlib Functions**

| File | Driverlib Function |
|---|---|
| **DXLOCK** | |
| memcfg.c | MemCfg_lockConfig |
| memcfg.c | MemCfg_unlockConfig |
| **DXCOMMIT** | |
| memcfg.c | MemCfg_commitConfig |
| **DXACCPROT0** | |
| memcfg.c | MemCfg_setProtection |
| **DXTEST** | |
| memcfg.c | MemCfg_setTestMode |
| **DXINIT** | |
| memcfg.c | MemCfg_initSections |
| memcfg.c | MemCfg_getInitStatus |
| **DXINITDONE** | |
| memcfg.c | MemCfg_getInitStatus |
| **LSXLOCK** | |
| memcfg.c | MemCfg_lockConfig |
| memcfg.c | MemCfg_unlockConfig |
| **LSXCOMMIT** | |
| memcfg.c | MemCfg_commitConfig |
| **LSXMSEL** | |
| memcfg.c | MemCfg_setLSRAMControllerSel |
| **LSXCLAPGM** | |
| memcfg.h | MemCfg_setCLAMemType |
| **LSXACCPROT0** | |
| memcfg.c | MemCfg_setProtection |
| **LSXACCPROT1** | |
| - | |
| **LSXTEST** | |
| memcfg.c | MemCfg_setTestMode |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-381. MEMCFG Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| **LSXINIT** | |
| memcfg.c | MemCfg_initSections |
| memcfg.c | MemCfg_getInitStatus |
| **LSXINITDONE** | |
| memcfg.c | MemCfg_getInitStatus |
| **GSXLOCK** | |
| memcfg.c | MemCfg_lockConfig |
| memcfg.c | MemCfg_unlockConfig |
| **GSXCOMMIT** | |
| memcfg.c | MemCfg_commitConfig |
| **GSXMSEL** | |
| memcfg.c | MemCfg_setGSRAMControllerSel |
| **GSXACCPROT0** | |
| memcfg.c | MemCfg_setProtection |
| **GSXACCPROT1** | |
| - | See GSXACCPROT0 |
| **GSXACCPROT2** | |
| - | See GSXACCPROT0 |
| **GSXACCPROT3** | |
| - | See GSXACCPROT0 |
| **GSXTEST** | |
| memcfg.c | MemCfg_setTestMode |
| **GSXINIT** | |
| memcfg.c | MemCfg_initSections |
| memcfg.c | MemCfg_getInitStatus |
| **GSXINITDONE** | |
| memcfg.c | MemCfg_getInitStatus |
| **MSGXTEST** | |
| memcfg.c | MemCfg_setTestMode |
| **MSGXINIT** | |
| memcfg.c | MemCfg_initSections |
| memcfg.c | MemCfg_getInitStatus |
| **MSGXINITDONE** | |
| memcfg.c | MemCfg_getInitStatus |
| **EMIF1LOCK** | |
| emif.h | EMIF_lockAccessConfig |
| emif.h | EMIF_unlockAccessConfig |
| **EMIF1COMMIT** | |
| emif.h | EMIF_commitAccessConfig |
| **EMIF1MSEL** | |
| emif.h | EMIF_selectController |
| **EMIF1ACCPROT0** | |
| emif.h | EMIF_setAccessProtection |
| **EMIF2LOCK** | |
| - | |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-381. MEMCFG Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| **EMIF2COMMIT** | |
| - | |
| **EMIF2ACCPROT0** | |
| - | |
| **NMAVFLG** | |
| memcfg.h | MemCfg_getViolationInterruptStatus |
| **NMAVSET** | |
| memcfg.h | MemCfg_forceViolationInterrupt |
| **NMAVCLR** | |
| memcfg.h | MemCfg_clearViolationInterruptStatus |
| **NMAVINTEN** | |
| memcfg.h | MemCfg_enableViolationInterrupt |
| memcfg.h | MemCfg_disableViolationInterrupt |
| **NMCPURDAVADDR** | |
| memcfg.c | MemCfg_getViolationAddress |
| **NMCPUWRAVADDR** | |
| memcfg.c | MemCfg_getViolationAddress |
| **NMCPUFAVADDR** | |
| - | |
| **NMDMAWRAVADDR** | |
| - | |
| **NMCLA1RDAVADDR** | |
| - | |
| **NMCLA1WRAVADDR** | |
| - | |
| **NMCLA1FAVADDR** | |
| - | |
| **MAVFLG** | |
| memcfg.h | MemCfg_getViolationInterruptStatus |
| **MAVSET** | |
| memcfg.h | MemCfg_forceViolationInterrupt |
| **MAVCLR** | |
| memcfg.h | MemCfg_clearViolationInterruptStatus |
| **MAVINTEN** | |
| memcfg.h | MemCfg_enableViolationInterrupt |
| memcfg.h | MemCfg_disableViolationInterrupt |
| **MCPUFAVADDR** | |
| memcfg.c | MemCfg_getViolationAddress |
| **MCPUWRAVADDR** | |
| - | |
| **MDMAWRAVADDR** | |
| - | |
| **UCERRFLG** | |
| memcfg.h | MemCfg_getUncorrErrorStatus |
| **UCERRSET** | |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-381. MEMCFG Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| memcfg.h | MemCfg_forceUncorrErrorStatus |
| **UCERRCLR** | |
| memcfg.h | MemCfg_clearUncorrErrorStatus |
| **UCCPUREADDR** | |
| memcfg.c | MemCfg_getUncorrErrorAddress |
| **UCDMAREADDR** | |
| memcfg.c | MemCfg_getUncorrErrorAddress |
| **UCCLA1READDR** | |
| - | |
| **CERRFLG** | |
| memcfg.h | MemCfg_getCorrErrorStatus |
| **CERRSET** | |
| memcfg.h | MemCfg_forceCorrErrorStatus |
| **CERRCLR** | |
| memcfg.h | MemCfg_clearCorrErrorStatus |
| **CCPUREADDR** | |
| memcfg.c | MemCfg_getCorrErrorAddress |
| **CERRCNT** | |
| memcfg.h | MemCfg_getCorrErrorCount |
| **CERRTHRES** | |
| memcfg.h | MemCfg_setCorrErrorThreshold |
| **CEINTFLG** | |
| memcfg.h | MemCfg_getCorrErrorInterruptStatus |
| **CEINTCLR** | |
| memcfg.h | MemCfg_clearCorrErrorInterruptStatus |
| **CEINTSET** | |
| memcfg.h | MemCfg_forceCorrErrorInterrupt |
| **CEINTEN** | |
| memcfg.h | MemCfg_enableCorrErrorInterrupt |
| memcfg.h | MemCfg_disableCorrErrorInterrupt |
| **ROMWAITSTATE** | |
| memcfg.h | MemCfg_enableROMWaitState |
| memcfg.h | MemCfg_disableROMWaitState |
| **ROMPREFETCH** | |
| memcfg.h | MemCfg_enableROMPrefetch |
| memcfg.h | MemCfg_disableROMPrefetch |

### 3.17.27.9 FLASH Registers to Driverlib Functions

**Table 3-382. FLASH Registers to Driverlib Functions**

| File | Driverlib Function |
|---|---|
| **FRDCNTL** | |
| flash.h | Flash_setWaitstates |
| **FBAC** | |
| flash.h | Flash_setBankPowerUpDelay |
| **FBFALLBACK** | |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-382. FLASH Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| flash.h | Flash_setBankPowerMode |
| **FBPRDY** | |
| flash.h | Flash_isBankReady |
| flash.h | Flash_isPumpReady |
| **FPAC1** | |
| flash.h | Flash_setPumpPowerMode |
| flash.h | Flash_setPumpWakeupTime |
| **FMSTAT** | |
| - | |
| **FRD_INTF_CTRL** | |
| flash.h | Flash_enablePrefetch |
| flash.h | Flash_disablePrefetch |
| flash.h | Flash_enableCache |
| flash.h | Flash_disableCache |
| **ECC_ENABLE** | |
| flash.h | Flash_enableECC |
| flash.h | Flash_disableECC |
| **SINGLE_ERR_ADDR_LOW** | |
| flash.h | Flash_getSingleBitErrorAddressLow |
| **SINGLE_ERR_ADDR_HIGH** | |
| flash.h | Flash_getSingleBitErrorAddressHigh |
| **UNC_ERR_ADDR_LOW** | |
| flash.h | Flash_getUncorrectableErrorAddressLow |
| **UNC_ERR_ADDR_HIGH** | |
| flash.h | Flash_getUncorrectableErrorAddressHigh |
| **ERR_STATUS** | |
| flash.h | Flash_getLowErrorStatus |
| flash.h | Flash_getHighErrorStatus |
| flash.h | Flash_clearLowErrorStatus |
| flash.h | Flash_clearHighErrorStatus |
| **ERR_POS** | |
| flash.h | Flash_getLowErrorPosition |
| flash.h | Flash_getHighErrorPosition |
| flash.h | Flash_getLowErrorType |
| flash.h | Flash_getHighErrorType |
| **ERR_STATUS_CLR** | |
| flash.h | Flash_clearLowErrorStatus |
| flash.h | Flash_clearHighErrorStatus |
| **ERR_CNT** | |
| flash.h | Flash_getErrorCount |
| **ERR_THRESHOLD** | |
| flash.h | Flash_setErrorThreshold |
| **ERR_INTFLG** | |
| flash.h | Flash_getInterruptFlag |
| **ERR_INTCLR** | |



Copyright © 2024 Texas Instruments Incorporated

**Table 3-382. FLASH Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| flash.h | Flash_clearSingleErrorInterruptFlag |
| flash.h | Flash_clearUncorrectableInterruptFlag |
| **FDATAH_TEST** | |
| flash.h | Flash_setDataHighECCTest |
| **FDATAL_TEST** | |
| flash.h | Flash_setDataLowECCTest |
| **FADDR_TEST** | |
| flash.h | Flash_setECCTestAddress |
| **FECC_TEST** | |
| flash.h | Flash_setECCTestECCBits |
| **FECC_CTRL** | |
| flash.h | Flash_enableECCTestMode |
| flash.h | Flash_disableECCTestMode |
| flash.h | Flash_selectLowECCBlock |
| flash.h | Flash_selectHighECCBlock |
| flash.h | Flash_performECCCalculation |
| **FOUTH_TEST** | |
| flash.h | Flash_getTestDataOutHigh |
| **FOUTL_TEST** | |
| flash.h | Flash_getTestDataOutLow |
| **FECC_STATUS** | |
| flash.h | Flash_getECCTestStatus |
| flash.h | Flash_getECCTestErrorPosition |
| flash.h | Flash_getECCTestSingleBitErrorType |
| **PUMPREQUEST** | |
| flash.h | Flash_claimPumpSemaphore |
| flash.h | Flash_releasePumpSemaphore |



Copyright © 2024 Texas Instruments Incorporated

Chapter 4

# ROM Code and Peripheral Booting

This chapter describes the booting functionality.

Further information about the boot-loading process can be found in the *TMS320F28004x Boot Features and Configurations Application Report*.


**4.1 Introduction** ..... 615
**4.2 Boot ROM Registers** ..... 615
**4.3 Device Boot Sequence** ..... 615
**4.4 Device Boot Modes** ..... 616
**4.5 Configuring Boot Mode Pins** ..... 617
**4.6 Configuring Get Boot Options** ..... 619
**4.7 Configuring Emulation Boot Options** ..... 620
**4.8 Device Boot Flow Diagrams** ..... 621
**4.9 Device Reset and Exception Handling** ..... 627
**4.10 Boot ROM Description** ..... 629





Copyright © 2024 Texas Instruments Incorporated

## 4.1 Introduction

This chapter explains the boot ROM code functionality including the boot procedure when executed, the functions and features of the boot ROM code, and details 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 application code. The boot ROM also contains peripheral bootloaders which can be used to load an application into RAM. ROM Memory is shown in Table 4-1.

**Table 4-1. ROM Memory**

| ROM | CPU1 Size | CPU2 Size |
|---|---|---|
| Unsecure boot ROM | 64KB | 64KB |
| Secure ROM | 64KB | 64KB |
| CLA Data ROM | 8KB | 8KB |

## 4.2 Boot ROM Registers

The boot process code accesses several memory addresses and registers during execution. There are two sets of addresses, one for emulation and one for standalone boot flow. The emulation boot-control locations emulate OTP configurations and can be written to as many times as needed. The user configurable DCSM OTP locations used in the standalone boot flow program the device OTP and thus can only be written once. Table 4-2 details these locations.

**Table 4-2. Boot ROM Registers**

| Boot Flow | Register Name | Boot ROM Name | Address |
|---|---|---|---|
| Emulation | - | EMU_BOOTCTRL | 0x0000 0D00 |
| Standalone (Using Z1) | Z1_BOOTCTRL | Z1_BOOTCTRL | 0x0005 F004 |
| Standalone (Using Z2) | Z2_BOOTCTRL | Z2_BOOTCTRL | 0x0005 F044 |

## 4.3 Device Boot Sequence

The boot sequence, Table 4-3, describes the general boot ROM procedure each time the CPU core is reset. For dual-core devices, CPU1 is the master controller and controls the boot process. Each CPU goes through a boot procedure, but under the control of CPU1. The exception to this rule is when CPU2 is set to boot-to-Flash, in which CPU1 is not involved.

During booting, the boot ROM code updates a boot status location in RAM that details the actions taken during this process. Refer to Section 4.10.11 for more details.

**Table 4-3. Boot ROM Sequence**

| Step | CPU1 Action | CPU2 Action |
|---|---|---|
| 1 | After reset, the FUSE error register is checked for any errors and are handled accordingly. | Held in reset. |
| 2 | Clock and Flash Configuration | Held in reset. |
| 3 | Device configuration registers are programmed from OTP. | Held in reset. |
| 4 | All CPU RAMs are initialized. | Held in reset. |
| 5 | Any pending NMI is handled by the code. | Held in reset. |
| 6 | DCSM initialization sequence is executed. | Held in reset. |
| 7 | Bring CPU2 out of reset. | Brought out of reset and performs:<br>1. Clock and Flash Configuration<br>2. All CPU RAMs are initialized<br>3. DCSM initialization |
| 8 | Based on the boot mode select GPIO pins and boot mode set in OTP, the boot mode is determined, and the appropriate boot sequence is executed. Refer to Section 4.8 for a flow chart of the device boot sequences. | Based on the boot mode value read from OTP, the appropriate boot sequence is executed. |



Copyright © 2024 Texas Instruments Incorporated

## 4.4 Device Boot Modes

This section explains the boot modes supported on this device. The boot ROM uses the boot select GPIO pins to determine the boot mode configuration. The device can be configured to boot to RAM, boot to Flash, execute a bootloader, or hold in a wait mode.

Table 4-4 shows the boot mode options available through selection by the default boot mode select pins. The boot mode select pins' GPIOs and realized boot mode for when Get boot mode is selected can be customized through the BOOTCTRL register detailed in Section 4.5.

**Table 4-4. Device Default Boot Modes for CPU1**

| Boot Mode | GPIO72 (Default boot mode select pin 1) | GPIO84 (Default boot mode select pin 0) |
|---|---|---|
| Parallel IO | 0 | 0 |
| SCI | 0 | 1 |
| Wait | 1 | 0 |
| Get / Flash(1) | 1 | 1 |

(1) Get boot mode, by default, on an unprogrammed device, or when the BOOTCTRL register contains an invalid key, boots to Flash mode. Get boot mode can be programmed on the device to change the default boot mode. Refer to Section 4.6 for more details on using Get boot mode.

**Table 4-5. All Available Boot Modes**

| Boot Mode | CPU Support |
|---|---|
| Parallel IO | CPU1 and CPU2 |
| SCI | CPU1 and CPU2 |
| Wait | CPU1 and CPU2 |
| Get | CPU1 and CPU2 |
| SPI | CPU1 and CPU2 |
| I2C | CPU1 and CPU2 |
| CAN | CPU1 and CPU2 |
| RAM | CPU1 and CPU2 |
| Flash | CPU1 and CPU2 |
| USB | CPU1 Only |

**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, meaning the SCI boot on the SCIA port. The same applies to the other peripheral boots.



Copyright © 2024 Texas Instruments Incorporated

## 4.5 Configuring Boot Mode Pins

This section details how the boot mode select pins can be customized by the user, by programming the BOOTCTRL register location in user-configurable DCSM OTP. The BOOTCTRL register, when programmed with a valid key, allows different GPIOs to be used for the two boot mode select pins. Additionally, the boot mode select pins allow the same GPIO to be assigned to each pin for single GPIO use cases. Also within the BOOTCTRL register, the default boot mode for use with Get boot can be changed. When debugging, EMU_BOOTCTRL is the emulation equivalent of the BOOTCTRL register and allows users to experiment with different boot modes without writing to OTP memory. Refer to Section 4.7 for details on values that can be set in the EMU_BOOTCTRL control word.

**Note**

Refer to Section 3.13 for the address of the BOOTCTRL register location in the user-configurable DCSM OTP.

**Table 4-6. BOOTCTRL Register Bit Fields for CPU1**

| Bit | Name | Description |
|---|---|---|
| 31-24 | Boot Mode Select Pin 1 (BMSP1) | Set to the GPIO pin to be used during boot (up to 255).<br>0 = Default BMSP1<br>1 = GPIO0<br>2 = GPIO1<br>...<br>255 = GPIO254 |
| 23-16 | Boot Mode Select Pin 0 (BMSP0) | Set to the GPIO pin to be used during boot (up to 255).<br>0 = Default BMSP0<br>1 = GPIO0<br>2 = GPIO1<br>...<br>255 = GPIO254 |
| 15-8 | Boot Mode (BMODE) | Boot mode definition when using Get boot mode option. Refer to Section 4.6 for valid BMODE values. |
| 7-0 | Key | Write 0x5A to these 8-bits to tell the boot ROM code that the bits in this register are valid |

**Table 4-7. BOOTCTRL Register Bit Fields for CPU2**

| Bit | Name | Description |
|---|---|---|
| 31-24 | Reserved | Reserved |
| 23-16 | Reserved | Reserved |
| 15-8 | Boot Mode (BMODE) | Boot mode definition when using Get boot mode option. Refer to Section 4.6 for valid BMODE values. |
| 7-0 | Key | Write 0x5A to these 8-bits to tell the boot ROM code that the bits in this register are valid |

On this device, the DCSM has two zones. Each zone, Z1 and Z2, has a copy of the BOOTCTRL register. The boot ROM is designed to be able to read from either location and uses the procedure in Figure 4-1 to identify which register to use. By default, if the Z1 BOOTCTRL is programmed, then that register is given the priority. If the Z1 BOOTCTRL is not programmed, then the boot ROM checks if Z2 BOOTCTRL is programmed; if not programmed, then the factory default options are used.



Copyright © 2024 Texas Instruments Incorporated

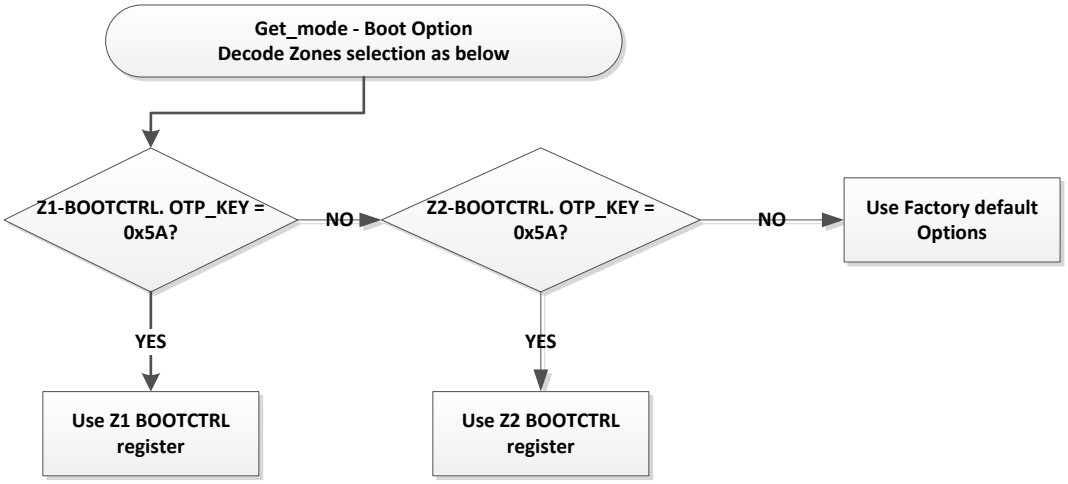



**Figure 4-1. Z1 and Z2 BOOTCTRL Selection**



Copyright © 2024 Texas Instruments Incorporated

## 4.6 Configuring Get Boot Options

In Get boot mode, the boot ROM reads the boot mode (BMODE) bit field in the BOOTCTRL register to determine which boot procedure to execute. By default, Get boot mode executes Flash boot when in standalone mode or wait boot mode when an emulator is connected to the device. Table 4-8 lists the values that can be set to the BMODE field in BOOTCTRL for CPU1 and the corresponding boot mode represented.Table 4-9 lists the acceptable values for CPU2. For additional details on the GPIOs used for each boot mode, refer to Section 4.10.6. When debugging the device using an emulator and EMU_BOOTCTRL, the BMODE field has some additional values that can be found in Section 4.7.

**Table 4-8. Get Mode Decoding on CPU1**

| Key | BMODE Value | Realized Boot Mode |
|---|---|---|
| != 0x5A | Don't Care | Flash Boot / Wait Boot(1) |
| = 0x5A | 0x00 | Parallel Boot |
| | 0x01 | SCI Boot 0 |
| | 0x02 | Wait Boot |
| | 0x04 | SPI Boot 0 |
| | 0x05 | I2C Boot 0 |
| | 0x07 | CAN Boot 0 |
| | 0x0A | RAM Boot |
| | 0x0B | Flash Boot |
| | 0x0C | USB Boot |
| | 0x81 | SCI Boot 1 |
| | 0x84 | SPI Boot 1 |
| | 0x85 | I2C Boot 1 |
| | 0x87 | CAN Boot 1 |
| | Other | Flash Boot / Wait Boot(1) |

(1) When an emulator is connected (TRSTn = 1) to the device, then an invalid EMU BOOTCTRL key or invalid EMU configured boot mode results in wait boot mode. If an emulator is connected with a valid EMU BOOTCTRL key and the EMU boot mode is configured to "Get Mode" boot then an invalid OTP BOOTCTRL key results in Flash boot mode. If an emulator is not connected with the boot mode selected to "Get Mode" boot, then an invalid OTP BOOTCTRL key or invalid OTP memory configured boot mode results in Flash boot mode.

**Table 4-9. Get Mode Decoding on CPU2**

| Key | BMODE Value | Realized Boot Mode |
|---|---|---|
| != 0x5A | Don't Care | Wait Boot |
| = 0x5A | 0x0A | Wait Boot / RAM Boot(1) |
| | 0x0B | Flash Boot |
| | Other | Wait Boot |

(1) Only after a hibernate reset, CPU2 can boot in RAM boot mode. After any other resets, CPU2 boots in wait boot mode.



Copyright © 2024 Texas Instruments Incorporated

## 4.7 Configuring Emulation Boot Options

When connected to the device using an emulator, the EMU_BOOTCTRL control word is used to determine the boot mode. This control word allows the user to experiment with various boot mode settings before writing to the BOOTCTRL register in the user-configurable DCSM OTP. The values that can be set in the BMODE field of the EMU_BOOTCTRL control word are listed in Section 4.7. Some notable options include being able to have emulation boot read from the boot mode select pins, emulate standalone boot using the values in OTP, and boot according to the Get boot value stored in OTP memory. Refer to Section 4.10.6 for details on the GPIOs used in the various boot modes.

**Note**

EMU_BOOTCTRL is not actually a register, but refers to a location in RAM (PIE RAM). PIE RAM starts at 0xD00, but the first few locations are reserved (when initializing the PIE vector table in application code) for these boot ROM variables.

**Table 4-10. Emulation Boot Options**

| Key | BMODE Value | Realized Boot Mode | CPU Support |
|---|---|---|---|
| != 0x5A | Don't Care | Wait Boot | CPU1 and CPU2 |
| = 0x5A | 0xFE | Boot as per BMSP0 and BMSP1 | CPU1 Only |
| | 0xFF | Emulate Standalone boot | CPU1 and CPU2 |
| | 0x00 | Parallel Boot | CPU1 and CPU2 |
| | 0x01 | SCI Boot 0 | CPU1 and CPU2 |
| | 0x02 | Wait Boot | CPU1 and CPU2 |
| | 0x03 | Get Mode(read OTP BOOTCTRL) | CPU1 and CPU2 |
| | 0x04 | SPI Boot 0 | CPU1 and CPU2 |
| | 0x05 | I2C Boot 0 | CPU1 and CPU2 |
| | 0x07 | CAN Boot 0 | CPU1 and CPU2 |
| | 0x0A | RAM Boot | CPU1 and CPU2 |
| | 0x0B | Flash Boot | CPU1 and CPU2 |
| | 0x0C | USB Boot | CPU1 Only |
| | 0x81 | SCI Boot 1 | CPU1 Only |
| | 0x84 | SPI Boot 1 | CPU1 Only |
| | 0x85 | I2C Boot 1 | CPU1 Only |
| | 0x87 | CAN Boot 1 | CPU1 Only |
| | Other | Wait Boot | CPU1 and CPU2 |



Copyright © 2024 Texas Instruments Incorporated

## 4.8 Device Boot Flow Diagrams

Figure 4-2 shows the device boot flow for CPU1 detailing the actions executed by boot ROM after a reset. Figure 4-3 shows the device boot flow for CPU2.

**Note**

For CPU1, the PLL is not used in the device clock path during boot, only INTOSC is used.

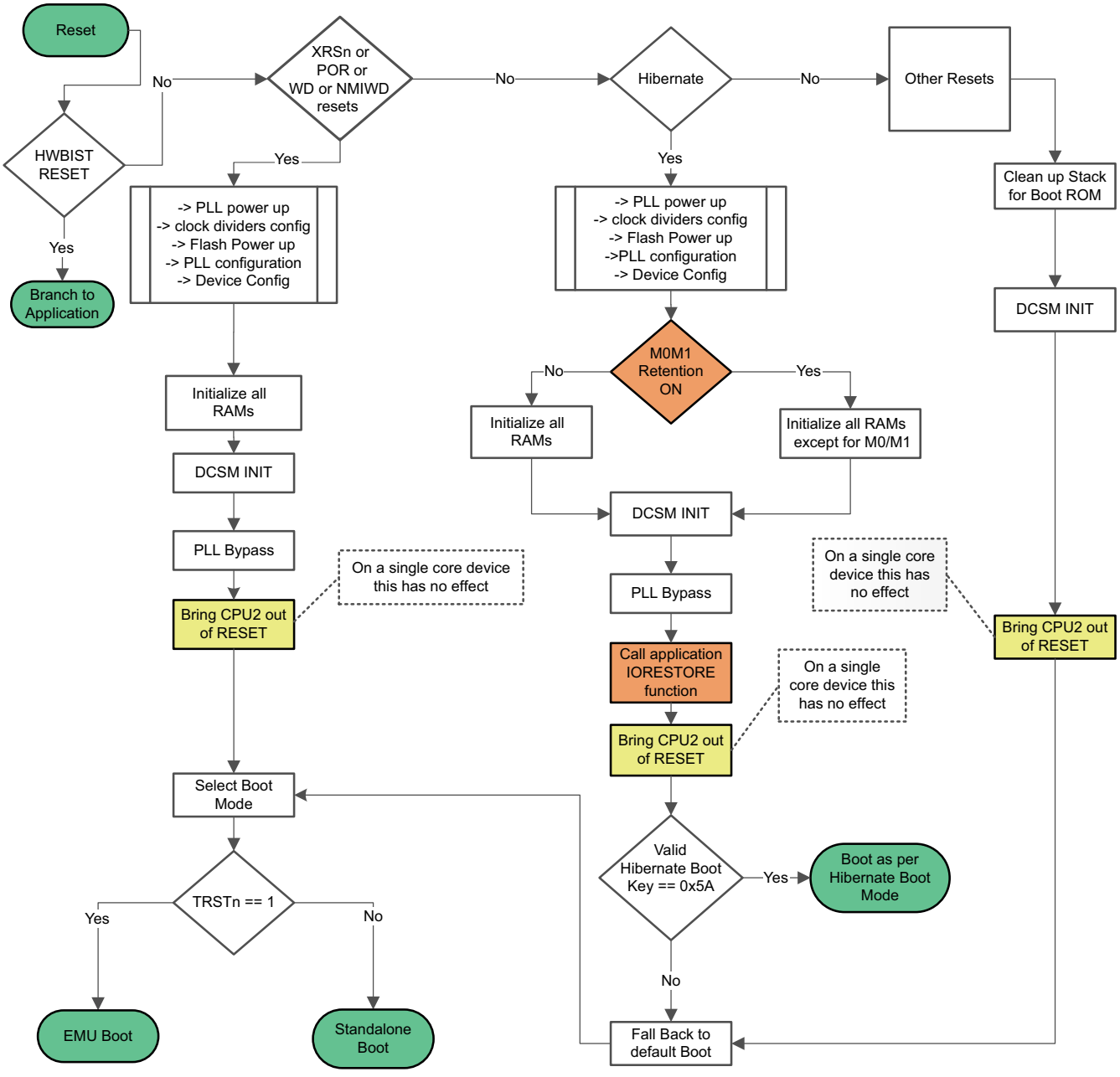



**Figure 4-2. CPU1 Device Boot Flow**



Copyright © 2024 Texas Instruments Incorporated

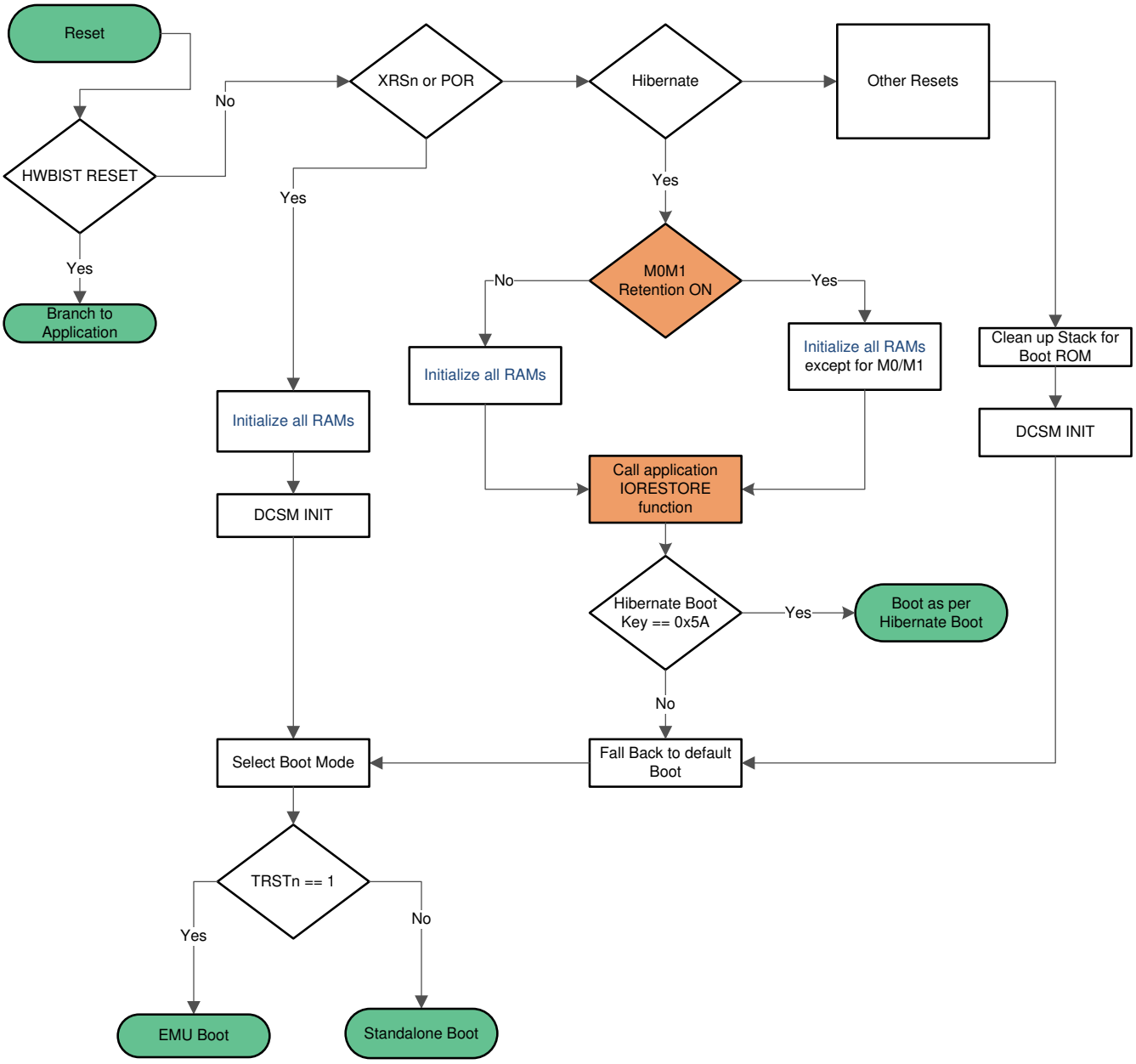



**Figure 4-3. CPU2 Device Boot Flow**



Copyright © 2024 Texas Instruments Incorporated

### 4.8.1 Emulation Boot Flow Diagrams

Figure 4-4 shows the boot flow for CPU1 when running the device in emulation mode. Figure 4-5 shows the emulation boot flow for CPU2.

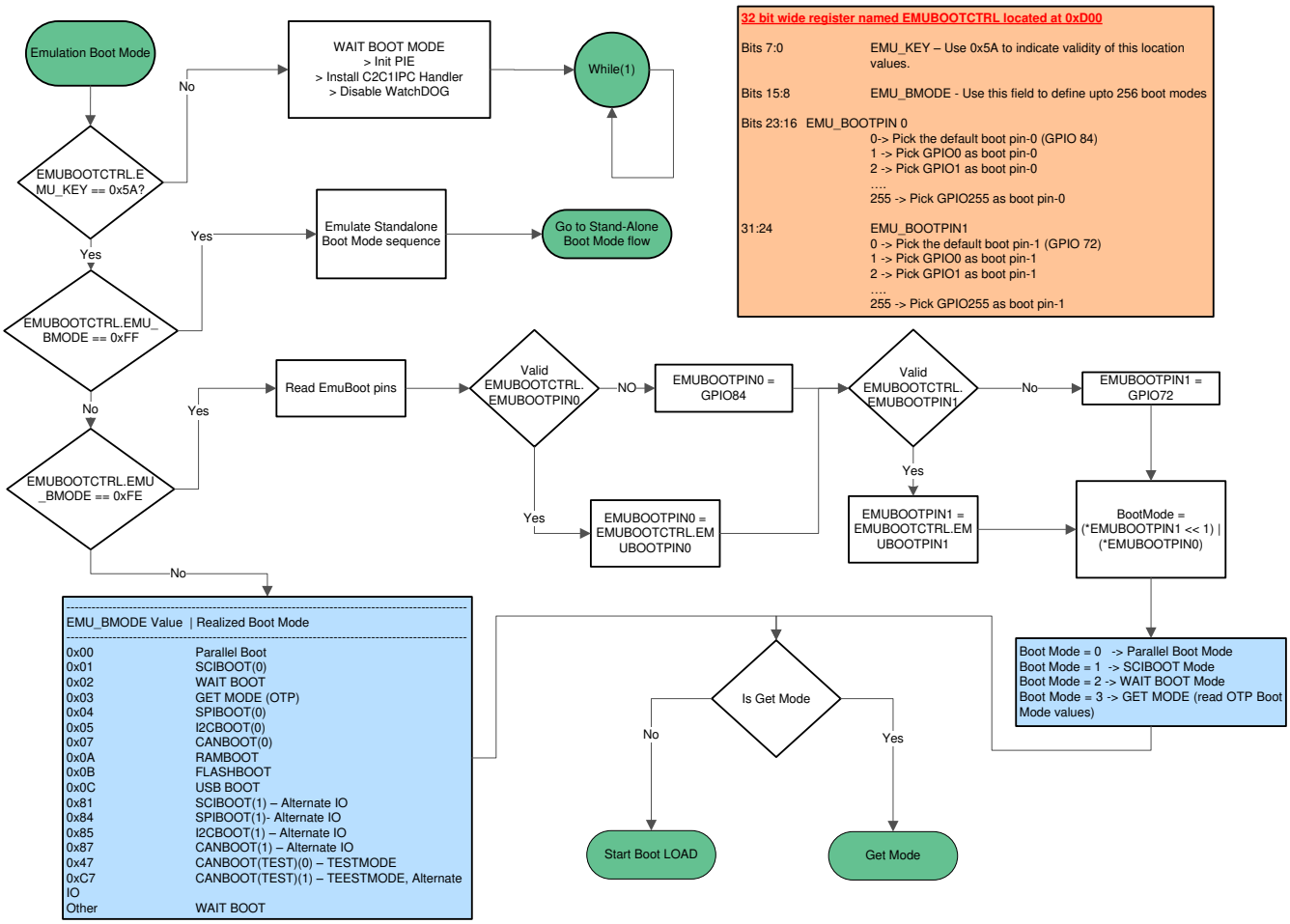



**Figure 4-4. CPU1 Emulation Boot Flow**



Copyright © 2024 Texas Instruments Incorporated

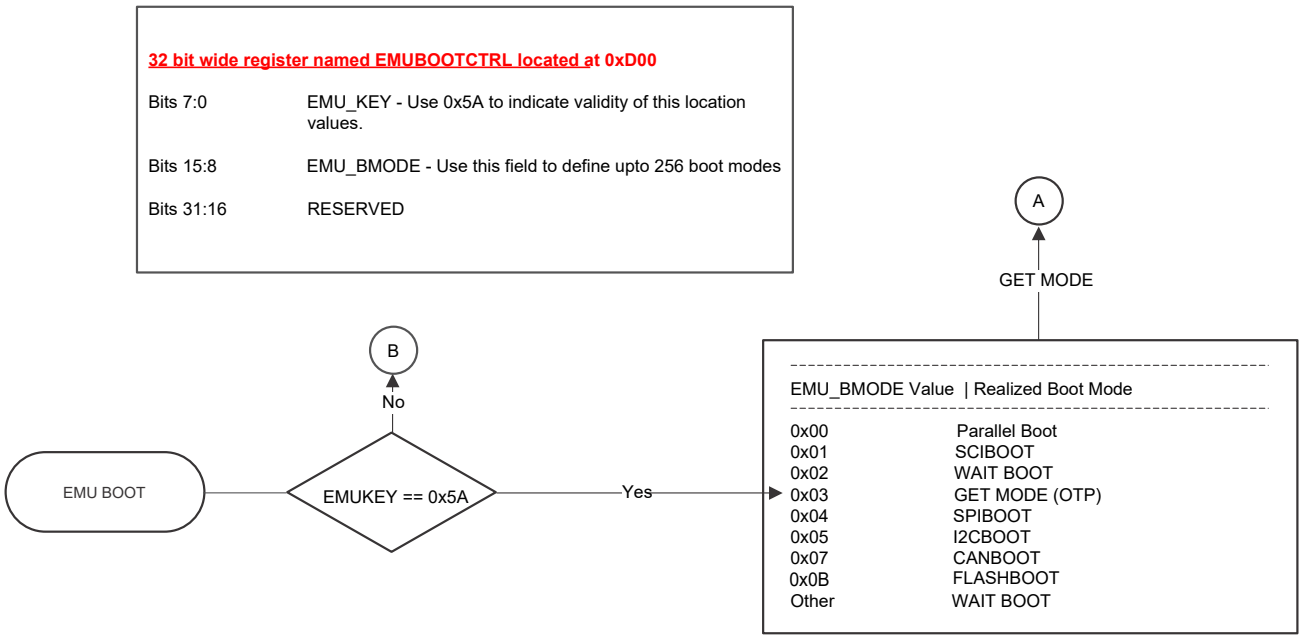



**Figure 4-5. CPU2 Emulation Boot Flow**



Copyright © 2024 Texas Instruments Incorporated

### 4.8.2 Standalone and Hibernate Boot Flow Diagrams

Figure 4-6 shows the device boot flow for CPU1 when running the device in standalone boot mode or when booting from hibernate. Figure 4-7 shows the standalone boot flow and hibernate boot flow for CPU2.

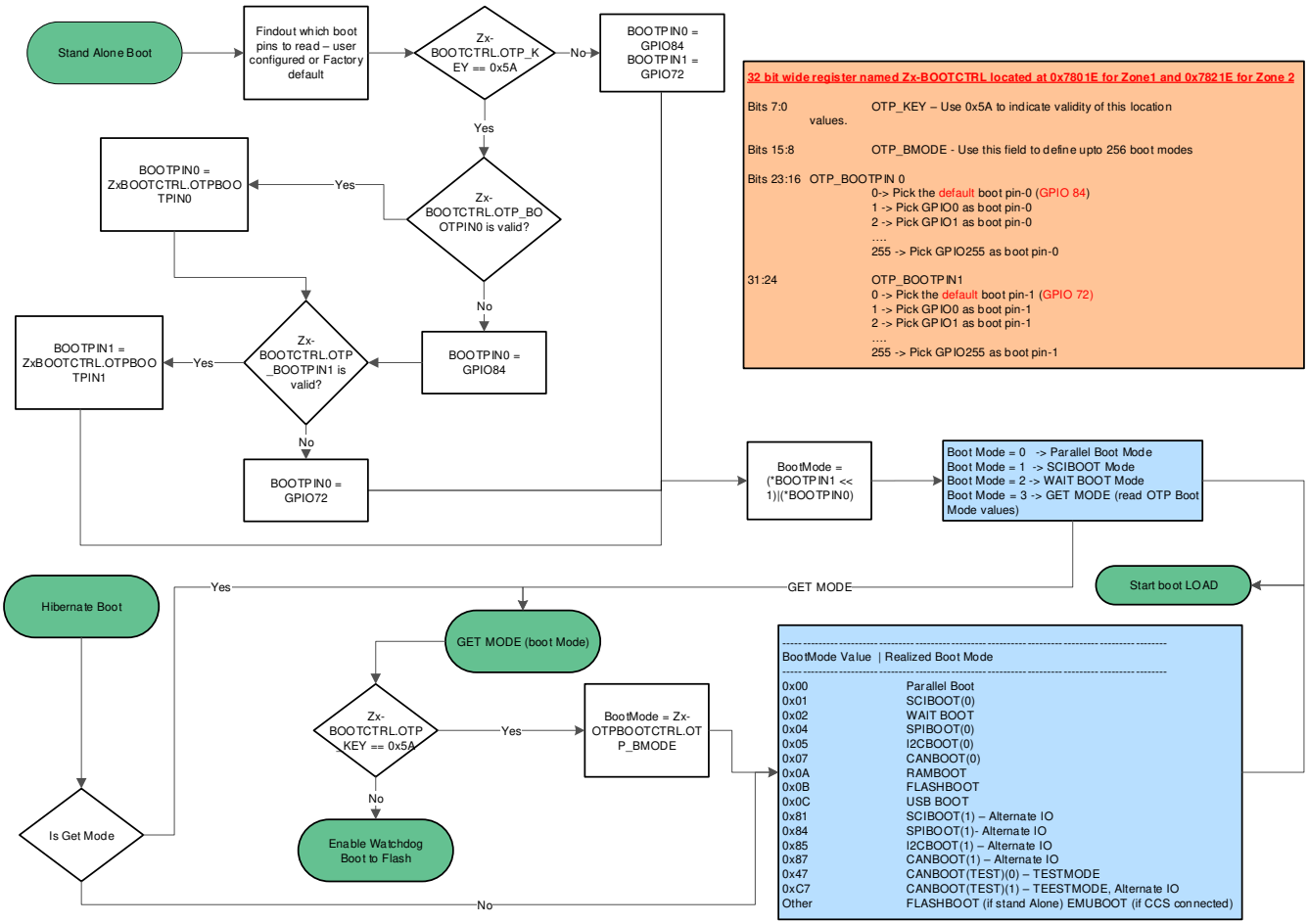



**Figure 4-6. CPU1 Standalone and Hibernate Boot Flow**



Copyright © 2024 Texas Instruments Incorporated

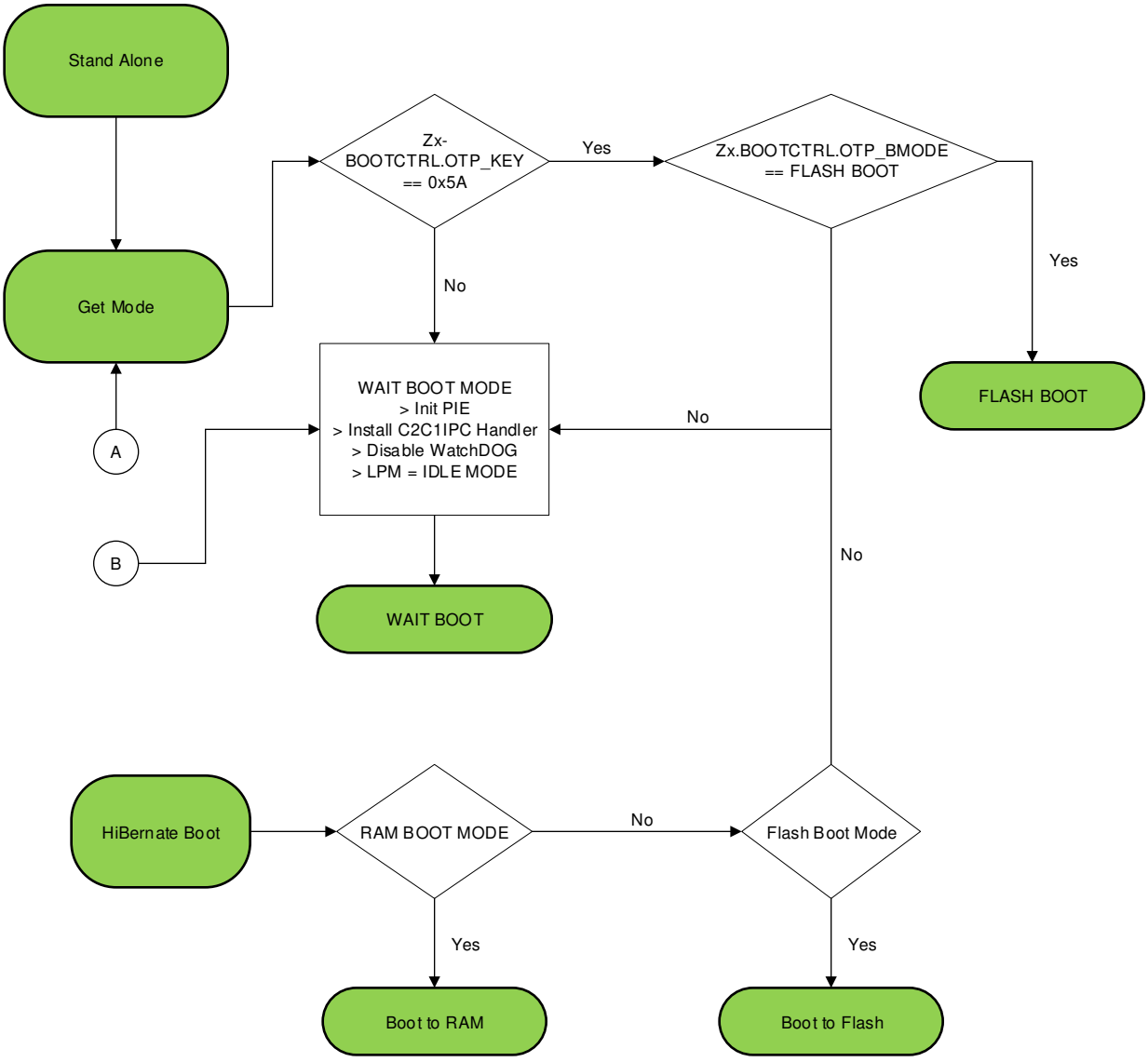



**Figure 4-7. CPU2 Standalone and Hibernate Boot Flow**



Copyright © 2024 Texas Instruments Incorporated

## 4.9 Device Reset and Exception Handling

### *4.9.1 Reset Causes and Handling*

This section explains the actions boot ROM performs upon reset after checking the reset cause.

**Table 4-11. Boot ROM Reset Causes and Actions**

| Reset Source | CPU1 Boot ROM Action | CPU2 Boot ROM Action |
|---|---|---|
| POR | 1. Adjust clock divider to /1<br>2. Device configuration<br>3. RAM initialization<br>4. Continue default boot flow | 1. RAM Initialization<br>2. Continue default boot flow |
| XRS | 1. Adjust clock divider to /1<br>2. Device configuration<br>3. RAM initialization<br>4. Continue default boot flow | 1. RAM Initialization<br>2. Continue default boot flow |
| HWBIST | Branch to application code | Branch to application code |
| Hibernate | 1. Adjust clock divider to /1<br>2. Device configuration<br>3. RAM initialization (Either all RAMS except for M0M1 or all RAMS)<br>4. Continue default boot flow | 1. RAM initialization (Either all RAMS except for M0M1 or all RAMS)<br>2. Continue default boot flow |
| WDRS (CPU1) | 1. Adjust clock divider to /1<br>2. Device configuration<br>3. RAM initialization<br>4. Continue default boot flow | 1. RAM initialization<br>2. Continue default boot flow |
| WDRS (CPU2) | Exception handled by CPU1 | 1. Clear boot stack<br>2. Continue default boot flow |
| NMIWDRS (CPU1) | 1. Adjust clock divider to /1<br>2. Device configuration<br>3. RAM initialization<br>4. Continue default boot flow | 1. RAM initialization<br>2. Continue default boot flow |
| NMIWDRS (CPU2) | Exception handled by CPU1 | 1. Clear boot stack<br>2. Continue default boot flow |
| Debugger (CPU1) | 1. Clear boot stack<br>2. Continue default boot flow | No Action |
| Debugger (CPU2) | Exception handled by CPU1 | 1. Clear boot stack<br>2. Continue default boot flow |
| SCCRESET (CPU1) | 1. Clear boot stack<br>2. Continue default boot flow | No Action |
| SCCRESET (CPU2) | Exception handled by CPU1 | 1. Clear boot stack<br>2. Continue default boot flow |



Copyright © 2024 Texas Instruments Incorporated

### 4.9.2 Exceptions and Interrupts Handling

This section explains the actions boot ROM performs if any exceptions that can occur happen during boot.

**Table 4-12. Boot ROM Exceptions and Actions**

| Exception Event Source | CPU1 Boot ROM Action | CPU2 Boot ROM Action | Event Logged |
|---|---|---|---|
| Single-bit error in FUSEERR | Ignore and continue to boot | Ignore and continue to boot | No |
| Multi-bit error in FUSEERR | Reset the device | No action required | No |
| Clock fail condition detected | Clear the NMI and continue to boot | Clear the NMI and continue to boot | Yes |
| Double-bit ECC error from RAM | Reset the device | Send IPC to CPU1, clear NMI flag, and wait forever | Yes |
| Double-bit error from Flash | Reset the device | Send IPC to CPU1, clear NMI flag, and wait forever | Yes |
| PIE Vector Error | Ignore and continue to boot | Ignore and continue to boot | Yes |
| HWBIST Error | Ignore and continue to boot | Send IPC to CPU1, clear NMI flag, and wait forever | Yes |
| ITRAP Exception | Provide ROM location where ROM loops | Provide ROM location to CPU1 of where ROM loops | Yes |
| CPU2 NMIWDRST | Ignore and continue to boot | No action required | Yes |
| CPU2 WDRST | Ignore and continue to boot | No action required | Yes |
| Any spurious PIE interrupt occurs | Acknowledge interrupt and continue to boot | Send IPC to CPU1 and acknowledge interrupt | No |



Copyright © 2024 Texas Instruments Incorporated

## 4.10 Boot ROM Description

This section explains the details regarding the device boot ROM.

### *4.10.1 Entry Points*

Table 4-13 gives the entry point addresses for various boot modes. These entry points indicate to the boot ROM where to branch to at the end of booting as per the selected boot mode.

**Table 4-13. Entry Point Addresses for CPU1 and CPU2**

| Entry Point | Address |
|---|---|
| RAM | 0x0000 0000 |
| Flash | 0x0008 0000 |

### *4.10.2 Wait Points*

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 4-14 gives the address ranges that the CPU1 PC register value falls between, if the boot code has entered one of these instances. Table 4-15 gives the addresses for CPU2.

**Table 4-14. Wait Point Addresses for CPU1**

| Address Range | Description |
|---|---|
| 0x003F E2D4 – 0x003F E2EF | In Wait Boot |
| 0x003F E73E – 0x003F E824 | In NMI Handler (startup) |
| 0x003F E35A – 0x003F E468 | In NMI Handler (PIE) |
| 0x003F E468 – 0x003F E495 | In ITRAP ISR |

**Table 4-15. Wait Point Addresses for CPU2**

| Address Range | Description |
|---|---|
| 0x003F E44C – 0x003F E451 | In Idle Mode |
| 0x003F DEAA – 0x003F DFC2 | In NMI Handler |
| 0x003F DE61 – 0x003F DEAA | In PIE Vector Mismatch Handler |
| 0x003F DFC2 – 0x003F E00B | In ITRAP ISR |



Copyright © 2024 Texas Instruments Incorporated

### 4.10.3 Memory Maps

This section details the ROM memory maps.

#### 4.10.3.1 CPU1 Boot ROM Memory Map

**Table 4-16. CPU1 Boot ROM Memory Map**

| Memory | Start Address | End Address | Length |
|---|---|---|---|
| ROM Signature | 0x003F 8000 | 0x003F 8001 | 0x0002 |
| TI-RTOS (ROM)(1) | 0x003F 8002 | 0x003F 9E0F | 0x1E0E |
| PLC Tables 1 | 0x003F 9E10 | 0x003F D817 | 0x3A08 |
| PLC Tables 2 | 0x003F D818 | 0x003F DE17 | 0x0600 |
| Boot | 0x003F DE18 | 0x003F FF31 | 0x211A |
| CRC Table | 0x003F FF32 | 0x003F FF39 | 0x0008 |
| BIST Signature | 0x003F FF3A | 0x003F FF79 | 0x0040 |
| Version | 0x003F FF7A | 0x003F FF7B | 0x0002 |
| Checksum | 0x003F FF7C | 0x003F FFBD | 0x0042 |
| Vectors | 0x003F FFBE | 0x003F FFFF | 0x0042 |
| TI-RTOS (Flash) | 0x0008 2000 | 0x0008 3FFF | 0x2000 |

(1) The SYS/BIOS (TI-RTOS) section in ROM is no longer supported and must not be used for new designs.

#### 4.10.3.2 CPU2 Boot ROM Memory Map

**Table 4-17. CPU2 Boot ROM Memory Map**

| Memory | Start Address | End Address | Length |
|---|---|---|---|
| ROM Signature | 0x003F 8000 | 0x003F 8001 | 0x0002 |
| TI-RTOS (ROM)(1) | 0x003F 8002 | 0x003F 9E0F | 0x1E0E |
| PLC Tables 1 | 0x003F 9E10 | 0x003F D817 | 0x3A08 |
| PLC Tables 2 | 0x003F D818 | 0x003F DE17 | 0x0600 |
| Boot | 0x003F DE18 | 0x003F FF31 | 0x211A |
| CRC Table | 0x003F FF32 | 0x003F FF39 | 0x0008 |
| BIST Signature | 0x003F FF3A | 0x003F FF79 | 0x0040 |
| Version | 0x003F FF7A | 0x003F FF7B | 0x0002 |
| Checksum | 0x003F FF7C | 0x003F FFBD | 0x0042 |
| Vectors | 0x003F FFBE | 0x003F FFFF | 0x0042 |
| TI-RTOS (Flash) | 0x0008 2000 | 0x0008 3FFF | 0x2000 |

(1) The SYS/BIOS (TI-RTOS) section in ROM is no longer supported and must not be used for new designs.

#### 4.10.3.3 CLA Data ROM Memory Map

**Table 4-18. CLA Data ROM Memory Map**

| Memory | Start Address | End Address | Length |
|---|---|---|---|
| FFT Tables (Load) | 0x0100 1070 | 0x0100 186F | 0x0800 |
| Data (Load) | 0x0100 1870 | 0x0100 1FF9 | 0x078A |
| Version (Load) | 0x0100 1FFA | 0x0100 1FFF | 0x0006 |



Copyright © 2024 Texas Instruments Incorporated

#### 4.10.3.4 Reserved RAM and Flash Memory-Map

**Table 4-19. Reserved RAM and Flash Memory-Map for CPU1**

| Memory | Description | Start Address | End Address | Length |
|---|---|---|---|---|
| RAM | Boot ROM | 0x0000 0002 | 0x0000 0122 | 0x0121 |
| | TI-RTOS(1) | 0x0000 0780 | 0x0000 07FF | 0x0080 |
| Flash | TI-RTOS(1) (2) | 0x0008 2000 | 0x0008 2823 | 0x0824 |

(1) If not planning on using TI-RTOS in ROM, then these memory locations are free to be used by the application.
(2) For using the TI-RTOS in Flash sector A, TI recommends that this sector be made unsecure, or at minimum, the sector must be verified that there is no secure zone claiming this sector.

**Table 4-20. Reserved RAM and Flash Memory-Map for CPU2**

| Memory | Description | Start Address | End Address | Length |
|---|---|---|---|---|
| RAM | Boot ROM | 0x0000 0002 | 0x0000 00A1 | 0x00A0 |
| | TI-RTOS(1) | 0x0000 0780 | 0x0000 07FF | 0x0080 |
| Flash | TI-RTOS(1) (2) | 0x0008 2000 | 0x0008 2823 | 0x0824 |

(1) If not planning to use TI-RTOS in ROM, then these memory locations are free to be used by the application.
(2) For using the TI-RTOS in Flash sector A, TI recommends that this sector be made unsecure, or at minimum, the sector must be verified that there is no secure zone claiming this sector.



Copyright © 2024 Texas Instruments Incorporated

#### 4.10.3.5 ROM Tables

This section details the boot ROM and CLA ROM symbol tables.

##### *4.10.3.5.1 Boot ROM Tables*

The boot ROM symbols and their addresses can be located in the .map file that is included with the released boot ROM source and header code. Within the .map file, locate the Global Symbols category to get a list of the boot ROM symbols and addresses present.

##### *4.10.3.5.2 CLA ROM Tables*

**Table 4-21. CLA Data ROM Tables**

| | Start Address From CLA in Hex | Start Address From CPU in Hex |
|---|---|---|
| _CLAatan2HalfPITable | F870 | 0100 1870 |
| _CLAINV2PI | F874 | 0100 1874 |
| _CLAatan2Table | F876 | 0100 1876 |
| _CLAasinHalfPITable | F9FC | 0100 19FC |
| _CLAatan2TableEnd | F9FC | 0100 19FC |
| _CLAasinTable | FA00 | 0100 1A00 |
| _CLAacosinHalfPITable | FB86 | 0100 1B86 |
| _CLAasinTableEnd | FB86 | 0100 1B86 |
| _CLAacosinTable | FB8A | 0100 1B8A |
| _CLAacosinTableEnd | FD0A | 0100 1D0A |
| _CLAsinTable | FD0A | 0100 1D0A |
| _CLAsincosTable | FD0A | 0100 1D0A |
| _CLAsincosTable_Sin0 | FD0A | 0100 1D0A |
| _CLAcosTable | FD4A | 0100 1D4A |
| _CLAsincosTable_Cos0 | FD4A | 0100 1D4A |
| _CLAsinTableEnd | FE0A | 0100 1E0A |
| _CLAcosTableEnd | FE4C | 0100 1E4C |
| _CLAsincosTable_TABLE_SIZE | FE4C | 0100 1E4C |
| _CLAsincosTable_TABLE_SIZEDivTwoPi | FE4E | 0100 1E4E |
| _CLAsincosTable_TwoPiDivTABLE_SIZE | FE50 | 0100 1E50 |
| _CLAsincosTable_TABLE_MASK | FE52 | 0100 1E52 |
| _CLAsincosTable_Coef0 | FE54 | 0100 1E54 |
| _CLAsincosTable_Coef1 | FE56 | 0100 1E56 |
| _CLAsincosTable_Coef1_pos | FE58 | 0100 1E58 |
| _CLAsincosTable_Coef2 | FE5A | 0100 1E5A |
| _CLAsincosTable_Coef3 | FE5C | 0100 1E5C |
| _CLAsincosTable_Coef3_neg | FE5E | 0100 1E5E |
| _CLALNV2 | FE60 | 0100 1E60 |
| _CLAsincosTableEnd | FE60 | 0100 1E60 |
| _CLALNVe | FE62 | 0100 1E62 |
| _CLALNV10 | FE64 | 0100 1E64 |
| _CLABIAS | FE66 | 0100 1E66 |
| _CLALN_TABLE_MASK1 | FE68 | 0100 1E68 |
| _CLALN_TABLE_MASK2 | FE6A | 0100 1E6A |
| _CLALnTable | FE6C | 0100 1E6C |
| _CLAINV1 | FF32 | 0100 1F32 |
| _CLALnTableEnd | FF32 | 0100 1F32 |



Copyright © 2024 Texas Instruments Incorporated

**Table 4-21. CLA Data ROM Tables (continued)**

| | Start Address From CLA in Hex | Start Address From CPU in Hex |
|---|---|---|
| _CLAINV2 | FF34 | 0100 1F34 |
| _CLAINV3 | FF36 | 0100 1F36 |
| _CLAINV4 | FF38 | 0100 1F38 |
| _CLAINV5 | FF3A | 0100 1F3A |
| _CLAINV6 | FF3C | 0100 1F3C |
| _CLAINV7 | FF3E | 0100 1F3E |
| _CLALOG10 | FF40 | 0100 1F40 |
| _CLAExpTable | FF42 | 0100 1F42 |
| _CLAExpTableEnd | FFF4 | 0100 1FF4 |
| CROM VERSION | FFFA | 0100 1FFA (2 16-bit words)<br>`.word 0x0100 ; Boot ROM Version v1.0`<br>`.word 0x0413 ; Month/Year: (ex: 0x0109 = 1/09 = Jan 2009)` |

### *4.10.4 Boot Modes*

The available boot modes supported on this device are detailed in this section. Each boot mode allows for various options, providing configurations with different IOs to be used, depending on the application.

While each subsection gives details regarding the implementation of the native boot modes on the device, the subsections do not address utilizing each boot mode for common system operations such as:

- Device Firmware Upgrade (DFU)
- Erasing the Flash memory
- Verifying the Flash memory
- Unlocking the security zones
- Running the embedded code from "main"
- Resetting the MCU

These operations and more are covered in the *Serial Flash Programming of C2000™ Microcontrollers Application Report*.

#### 4.10.4.1 Wait Boot Mode

The wait boot mode puts the CPU in a loop and does not branch to the user application code. The device can enter wait boot mode either manually or because an error occurred during boot up. TI recommends using wait boot when using a debugger to avoid any JTAG complications.

Actions resulting in entering wait boot mode:

- Wait boot is set by the user as the boot mode
- The boot mode is unrecognized and a debugger is connected to the device
- The emulation BOOCTRL key is not equal to 0xA5 or 0x5A
- An error occurs during emulation boot and the boot mode pins are decoded with a value not recognized as a valid boot mode



Copyright © 2024 Texas Instruments Incorporated

#### 4.10.4.2 SCI Boot Mode

The SCI boot mode asynchronously transfers code from SCI-A to internal memory. This boot mode only supports an incoming 8-bit data stream and follows the data flow as outlined in Example 4-1.

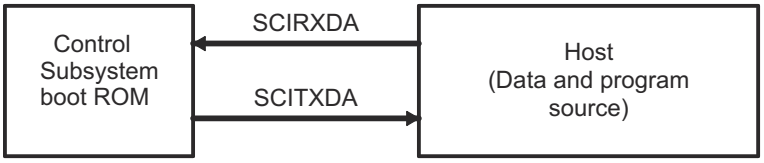



**Figure 4-8. Overview of SCI Bootloader Operation**

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.



Copyright © 2024 Texas Instruments Incorporated

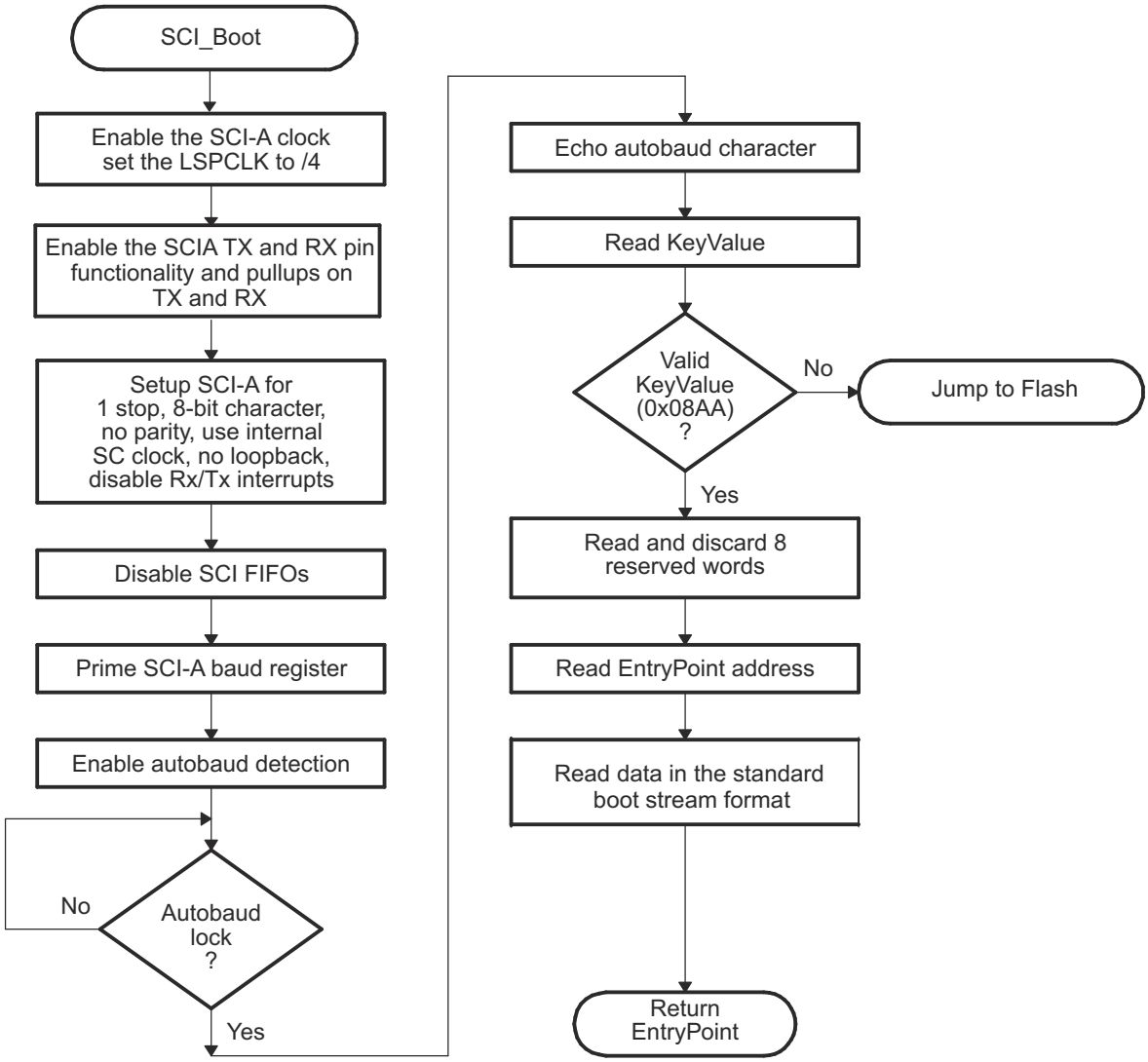



**Figure 4-9. Overview of SCI Boot Function**



Copyright © 2024 Texas Instruments Incorporated

#### 4.10.4.3 SPI Boot Mode

The SPI loader expects an SPI-compatible 16-bit or 24-bit addressable serial EEPROM or serial Flash device to be present on the SPI-A pins as indicated in Figure 4-10. The SPI bootloader supports an 8-bit data stream and does not support a 16-bit data stream.

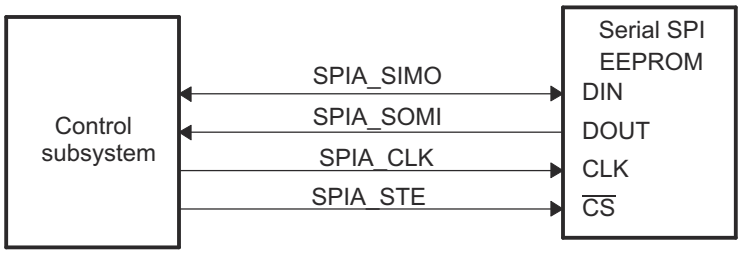



**Figure 4-10. SPI Loader**

The SPI boot ROM loader initializes the SPI module to interface to a serial SPI EEPROM or Flash. Devices of this type include, but are not limited to, the Xicor X25320 (4Kx8) and Xicor X25256 (32Kx8) SPI serial SPI EEPROMs and the Atmel AT25F1024A Serial Flash.

The SPI boot ROM loader initializes the SPI with the following settings: FIFO enabled, 8-bit character, internal SPICLK master mode and talk mode, clock phase = 1, polarity = 0, using the slowest baud rate.

If the download is to be performed from an SPI port on another device, then that device must be setup to operate in the slave mode and mimic a serial SPI EEPROM. Immediately after entering the SPI_Boot function, the pin functions for the SPI pins are set to primary and the SPI is initialized. The initialization is done at the slowest speed possible. Once the SPI is initialized and the key value read, you can specify a change in baud rate or low-speed peripheral clock.

**Table 4-22. SPI 8-Bit Data Stream**

| Byte | Contents |
|---|---|
| 1 | LSB: AA (KeyValue for memory width = 8 bits) |
| 2 | MSB: 08h (KeyValue for memory width = 8 bits) |
| 3 | LSB: LOSPCP |
| 4 | MSB: SPIBRR |
| 5 | LSB: reserved for future use |
| 6 | MSB: reserved for future use |
| ... | ... |
| ... | Data for this section. |
| 17 | LSB: reserved for future use |
| 18 | MSB: reserved for future use |
| 19 | LSB: Upper half (MSW) of Entry point PC[23:16] |
| 20 | MSB: Upper half (MSW) of Entry point PC[31:24] (Note: Always 0x00) |
| 21 | LSB: Lower half (LSW) of Entry point PC[7:0] |
| 22 | MSB: Lower half (LSW) of Entry point PC[15:8] |
| ... | .... |
| ... | Data for this section. |
| ... | Blocks of data in the format size/destination address/data as shown in the generic data stream description |
| ... | ... |
| ... | Data for this section. |
| n | LSB: 00h |
| n+1 | MSB: 00h - indicates the end of the source |



Copyright © 2024 Texas Instruments Incorporated

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 GPIO19 (SPISTE) pin 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 2 bytes fetched can be used to change the value of the low speed peripheral clock register (LOSPCP) and the SPI baud rate register (SPIBRR). The first byte read is the LOSPCP value and the second byte read is the SPIBRR value. The next 7 words are reserved for future enhancements. The SPI bootloader reads these 7 words and discards them.
7. The next two words makeup the 32-bit entry point address where execution can continue 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.

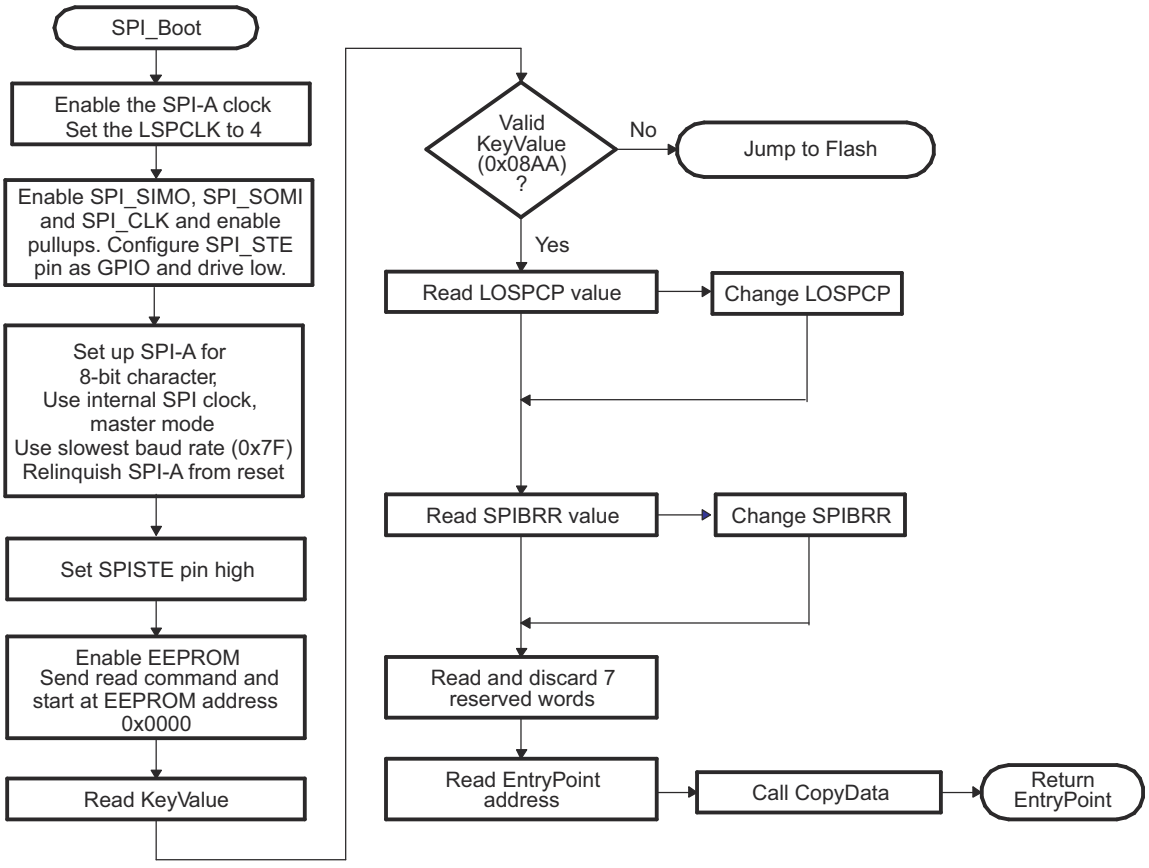



**Figure 4-11. Data Transfer From EEPROM Flow**



Copyright © 2024 Texas Instruments Incorporated

#### 4.10.4.4 I2C Boot Mode

The I2C bootloader expects an 8-bit wide I2C-compatible EEPROM device to be present at address 0x50 on the I2C-A bus as indicated in Figure 4-12. The EEPROM must adhere to conventional I2C EEPROM protocol, as described in this section, with a 16-bit base address architecture.

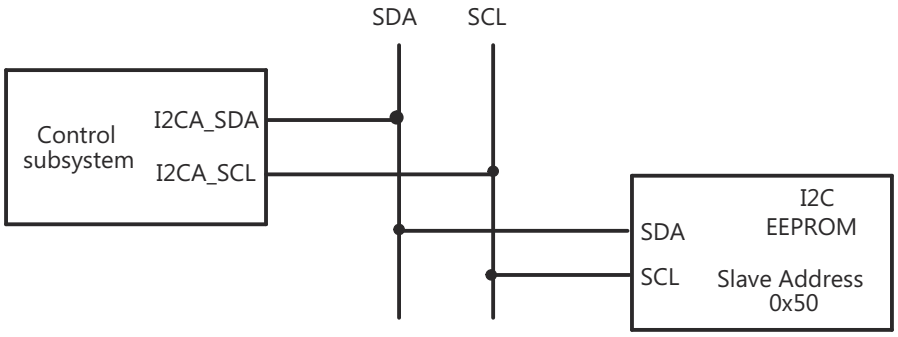



**Figure 4-12. EEPROM Device at Address 0x50**

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 application 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 4-23 shows the 8-bit data stream used by the I2C.



Copyright © 2024 Texas Instruments Incorporated

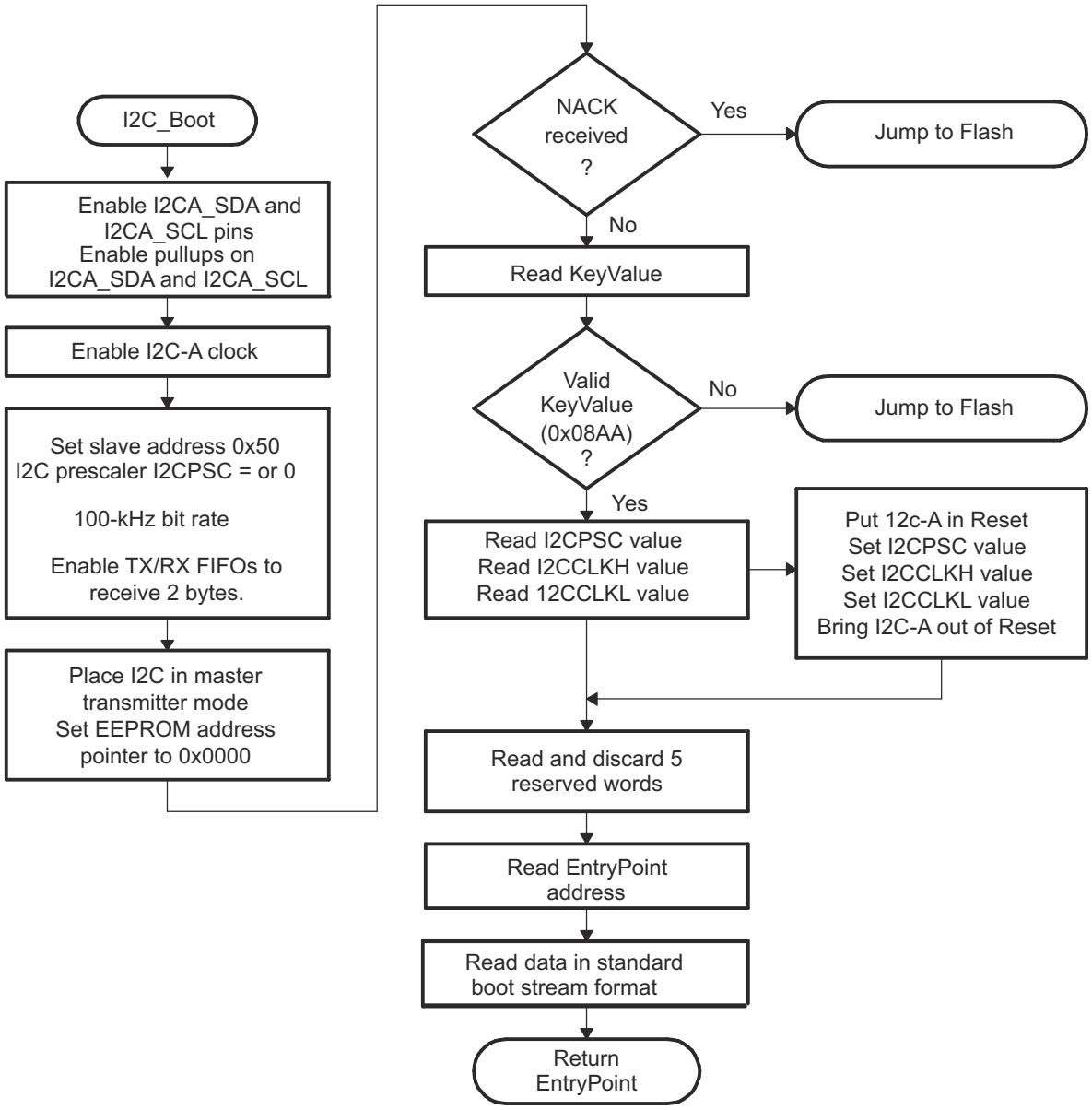



**Figure 4-13. Overview of I2C Boot Function**



Copyright © 2024 Texas Instruments Incorporated

**Table 4-23. I2C 8-Bit Data Stream**

| Byte | Contents |
|---|---|
| 1 | LSB: AA (KeyValue for memory width = 8 bits) |
| 2 | MSB: 08h (KeyValue for memory width = 8 bits) |
| 3 | LSB: I2CPSC[7:0] |
| 4 | reserved |
| 5 | LSB: I2CCLKH[7:0] |
| 6 | MSB: I2CCLKH[15:8] |
| 7 | LSB: I2CCLKL[7:0] |
| 8 | MSB: I2CCLKL[15:8] |
| ... | ... |
| ... | Data for this section. |
| | ... |
| 17 | LSB: Reserved for future use |
| 18 | MSB: Reserved for future use |
| 19 | LSB: Upper half of entry point PC |
| 20 | MSB: Upper half of entry point PC[22:16] (Note: Always 0x00) |
| 21 | LSB: Lower half of entry point PC[15:8] |
| 22 | MSB: Lower half of entry point PC[7:0] |
| ... | ... |
| ... | Data for this section. |
| | ... |
| | Blocks of data in the format size/destination address/data as shown in the generic data stream description. |
| ... | ... |
| ... | Data for this section. |
| | ... |
| n | LSB: 00h |
| n+1 | MSB: 00h - indicates the end of the source |

The I2C EEPROM protocol required by the I2C bootloader is shown in Figure 4-14 and Figure 4-15. The first communication, which sets the EEPROM address pointer to 0x0000 and reads the KeyValue (0x08AA), is shown in Figure 4-14. All subsequent reads are shown in Figure 4-15 and are read two bytes at a time.

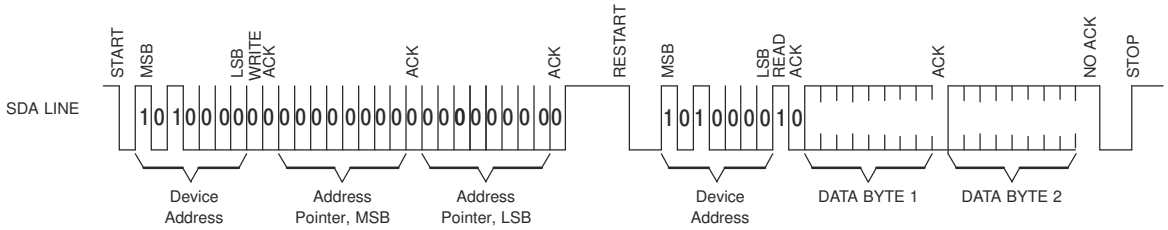



**Figure 4-14. Random Read**

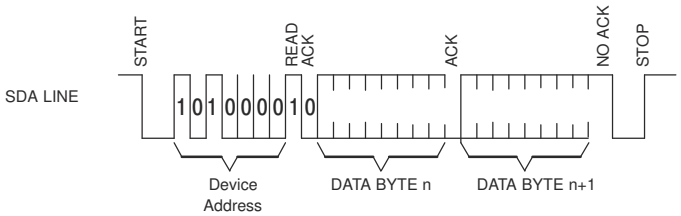



**Figure 4-15. Sequential Read**



Copyright © 2024 Texas Instruments Incorporated

#### 4.10.4.5 Parallel Boot Mode

The parallel general purpose I/O (GPIO) boot mode asynchronously transfers code from GPIO58-GPIO63, GPIO64-GPIO65 to internal memory. Each value is 8 bits long and follows the same data flow as outlined in Figure 4-16.

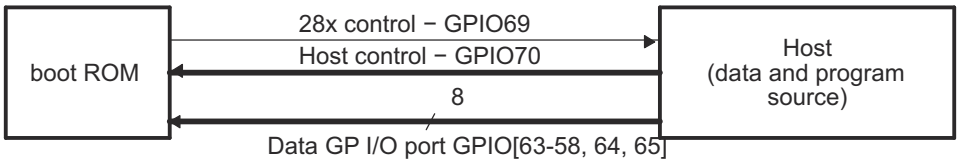



**Figure 4-16. Overview of Parallel GPIO Bootloader Operation**

The control subsystem communicates with the external host device by polling/driving the GPIO70 and GPIO69 lines. The handshake protocol shown in Figure 4-17 must be used to successfully transfer each word using GPIO[63-58,64,65]. This protocol is very robust and allows for a slower or faster host to communicate with the master subsystem.

Two consecutive 8-bit words are read to form a single 16-bit word. The most significant byte (MSB) is read first followed by the least-significant byte (LSB). In this case, data is read from GPIO[63-58,64,65].

The 8-bit data stream is shown in Table 4-24.

**Table 4-24. Parallel GPIO Boot 8-Bit Data Stream**

| Bytes | | GPIO[63:58,64,65] (Byte 1 of 2) | GPIO[63:58,64,65] (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 = 0x00BB 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 first block Addr[15:0] (Addr = 0xAABB CCDD) |
| 29 | 30 | BB | AA | First word of the first block in the source being loaded = 0xAABB |
| ...<br>... | | | | ...<br>Data for this section.<br>... |
| . | | 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<br>(More sections if required) |
| n+2 | n+3 | 00 | 00 | Block size of 0000h - indicates end of the source program |



Copyright © 2024 Texas Instruments Incorporated

The device first signals the host that the device is ready to begin data transfer by pulling the GPIO69 pin low. The host load then initiates the data transfer by pulling the GPIO70 pin low. The complete protocol is shown in Figure 4-17.

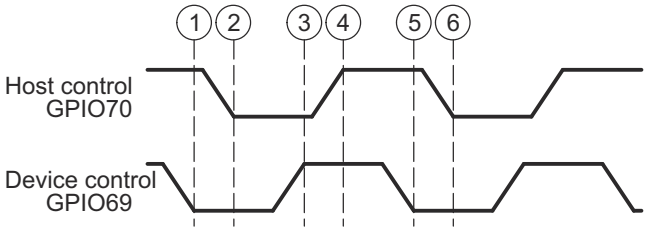



**Figure 4-17. Parallel GPIO Bootloader Handshake Protocol**

1. The device indicates the device is ready to start receiving data by pulling the GPIO69 pin low.
2. The bootloader waits until the host puts data on GPIO [63-58,64,65]. The host signals to the device that data is ready by pulling the GPIO70 pin low.
3. The device reads the data and signals the host that the read is complete by pulling GPIO69 high.
4. The bootloader waits until the host acknowledges the device by pulling GPIO70 high.
5. The device again indicates the device is ready for more data by pulling the GPIO69 pin low.

This process is repeated for each data value to be sent.

Figure 4-18 shows an overview of the Parallel GPIO bootloader flow.

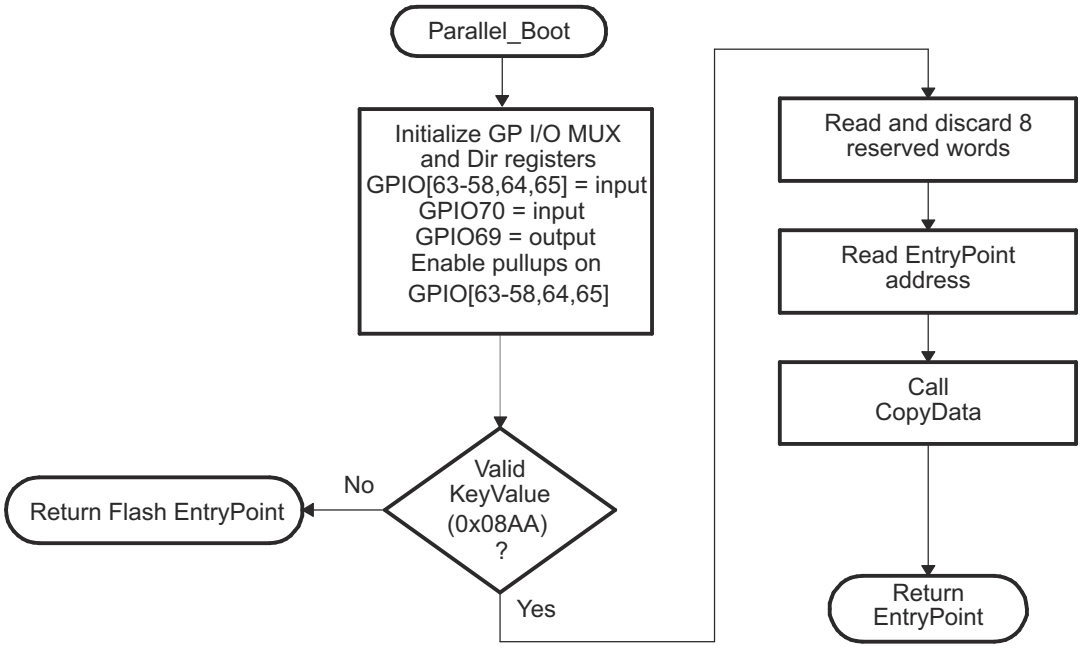



**Figure 4-18. Parallel GPIO Mode Overview**

Figure 4-19 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 waits for the host. In this manner, the protocol works with both a host running faster and a host running slower than the device.



Copyright © 2024 Texas Instruments Incorporated

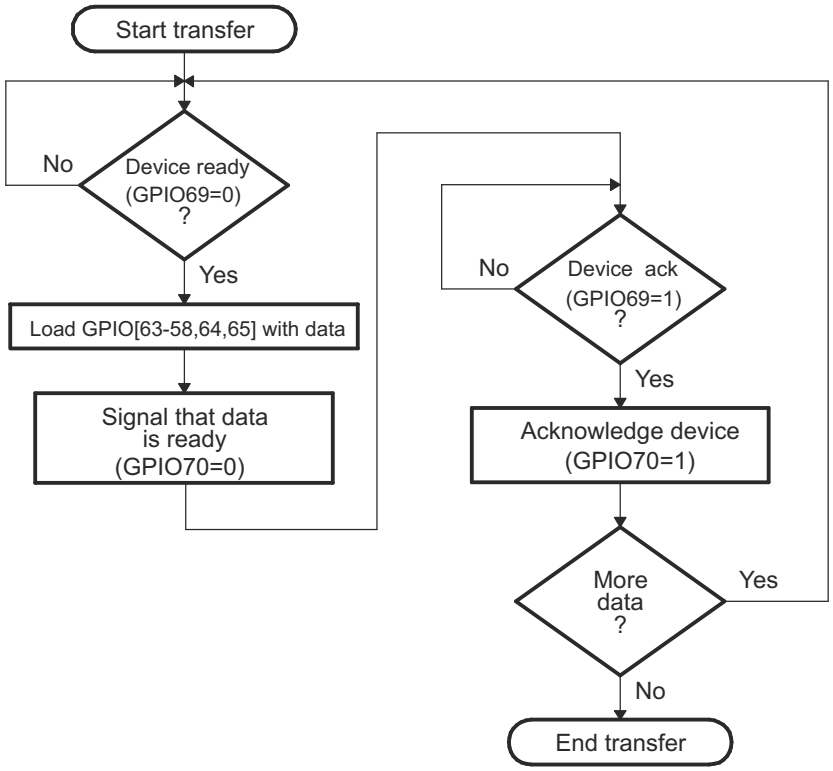



**Figure 4-19. Parallel GPIO Mode - Host Transfer Flow**

Figure 4-20 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 4-20, discards the upper 8 bits of the first read from the port and treats the lower 8 bits masked with GPIO65 in bit position 7 and GPIO64 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 combines the MSB and LSB into a single 16-bit value to be passed back to the calling routine.



Copyright © 2024 Texas Instruments Incorporated

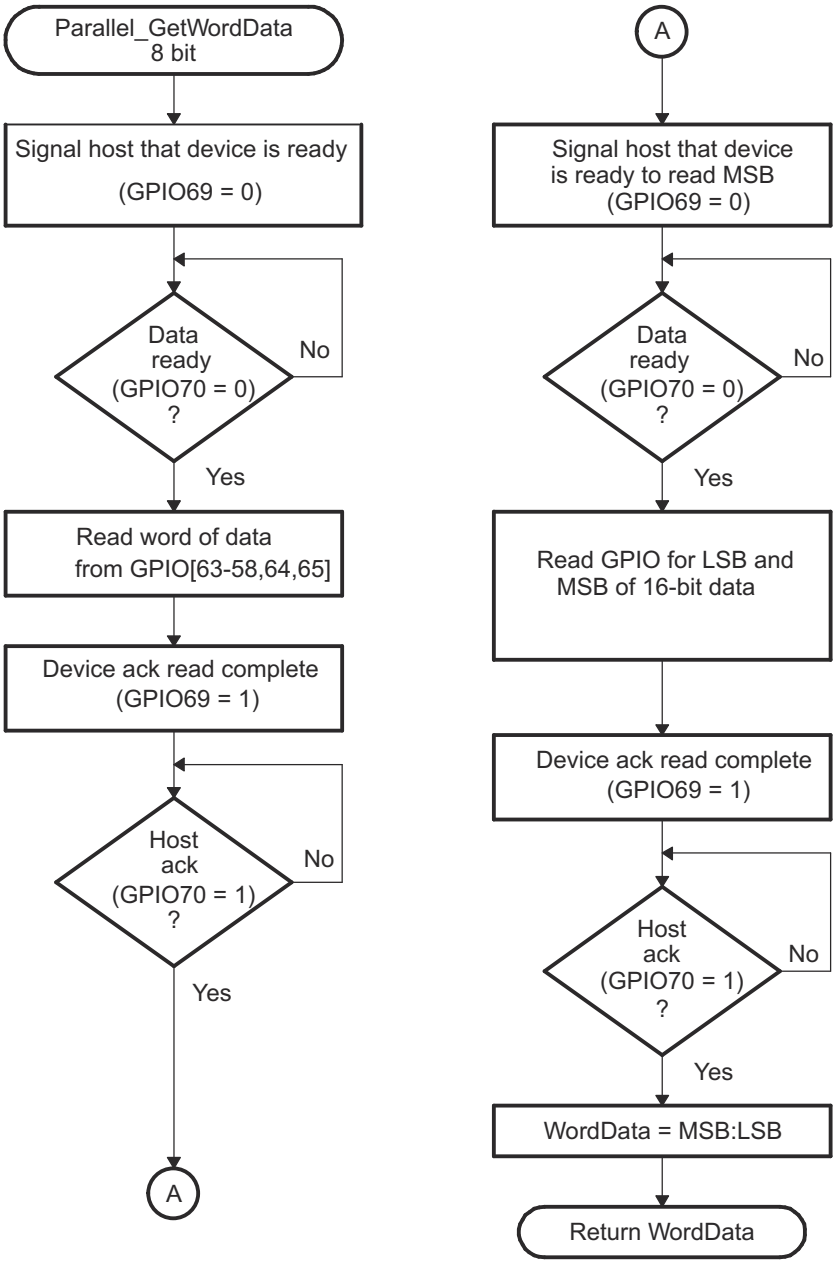



**Figure 4-20. 8-Bit Parallel GetWord Function**



Copyright © 2024 Texas Instruments Incorporated

#### 4.10.4.6 CAN Boot Mode

The CAN bootloader asynchronously transfers code from CAN-A to internal memory. The host can be any CAN node. The communication is first done with 11-bit standard identifiers (with a MSGID of 0x1) using two bytes per data frame. The host can download a kernel to reconfigure the CAN if higher data throughput is desired.

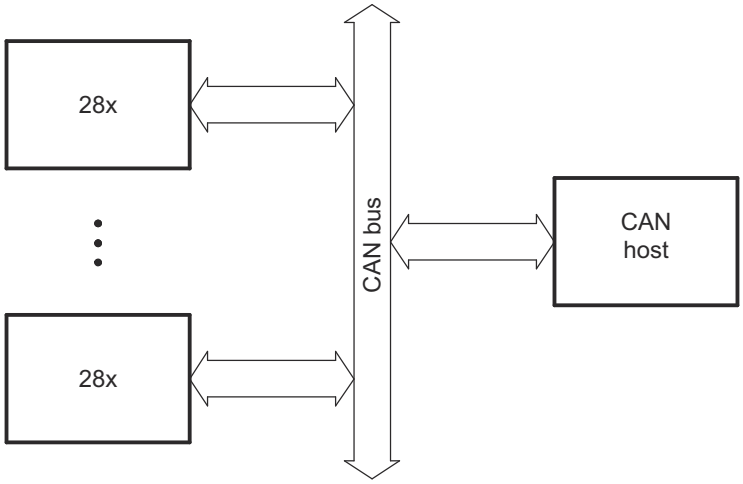



**Figure 4-21. Overview of CAN-A Bootloader Operation**

The bit timing registers are programmed in such a way that a 50kbps bit rate is achieved with the 10MHz INTOSC1 oscillator, a shown in Table 4-25.

**Table 4-25. Bit-Rate Value for Internal Oscillators**

| OSCCLK | SYSCLKOUT | Bit Rate |
|---|---|---|
| 10MHz | 10MHz | 50kbps |

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 CPU1 CAN boot loader uses INTOSC1 as the bit clock source and INTOSC2 as the system clock source. The CPU2 CAN boot loader does not change either clock source, so CPU1 must configure the clock sources before starting the CPU2 CAN boot loader.

Mailbox 1 is programmed with a standard MSGID of 0x1 for boot-loader communication. The CAN host can transmit only 2 bytes at a time, LSB first and MSB next. For example, to transmit the word 0x08AA to the device, transmit AA first, followed by 08. The program flow of the CAN bootloader is identical to the SCI bootloader. The data sequence for the CAN bootloader is shown in Table 4-26.



Copyright © 2024 Texas Instruments Incorporated

**Table 4-26. CAN 8-Bit Data Stream**

| Bytes | | Byte 1 of 2 | Byte 2 of 2 | Description |
|---|---|---|---|---|
| 1 | 2 | AA | 08 | 0x08AA (KeyValue for memory width = 16 bits) |
| 3 | 4 | 00 | 00 | reserved |
| 5 | 6 | 00 | 00 | reserved |
| 7 | 8 | 00 | 00 | reserved |
| 9 | 10 | 00 | 00 | reserved |
| 11 | 12 | 00 | 00 | reserved |
| 13 | 14 | 00 | 00 | reserved |
| 15 | 16 | 00 | 00 | reserved |
| 17 | 18 | 00 | 00 | reserved |
| 19 | 20 | BB | AA | Entry point PC[22:16] |
| 21 | 22 | DD | CC | Entry point PC[15:0] (PC = 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 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 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<br>(More sections if required) |
| n+2 | n+3 | 00 | 00 | Block size of 0000h - indicates end of the source program |



Copyright © 2024 Texas Instruments Incorporated

#### 4.10.4.7 USB Boot Mode

In USB boot mode, the device enumerates with vendor ID 0x1CBE and product ID 0x00FF. The device descriptor and interface descriptor both show the class as 0xFF (vendor-specific), the subclass as 0x00, and the protocol as 0x00. After enumeration, the device waits for data. Data can be sent using bulk OUT transfers to endpoint 1. The data is interpreted as a series of 8-bit bytes in the standard data stream format described in Section 4.10.5, shown here in Table 4-27. No reserved bytes are used. Once the data transfer is complete (block size of 0x0000 sent), the device disconnects from the USB bus, allowing other software to make use of the module if desired. Figure 4-22 illustrates the flow for USB boot mode.

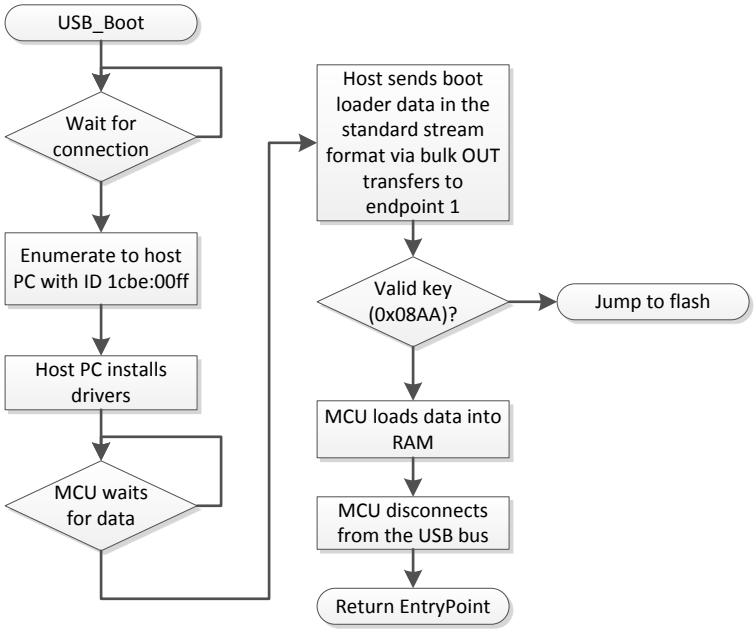



**Figure 4-22. USB Boot Flow**

Implementing PC-side USB software is not trivial. It is recommended to use the TI-provided tools and drivers to load data in USB boot mode. Hex and binary files for loader tools can be generated from COFF (.out) files using the hex2000 tool. To produce a plain binary file in the boot loader format, use the following command line:

hex2000 -boot -b Program_to_Load.out -o Binary_Loader_Data.dat

For more information on hex2000, see the *TMS320C28x Assembly Language Tools User's Guide*.

**Note**

INTOSC2 must be enabled before invoking the USB boot loader. If INTOSC2 is not enabled, the boot loader hangs. A debugger reset or SCC reset does not enable INTOSC2, if INTOSC2 has been disabled by the application.



Copyright © 2024 Texas Instruments Incorporated

**Table 4-27. USB 8-Bit Data Stream**

| Bytes | | First Byte (LSB) | Second Byte (MSB) | 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 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 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<br>(More sections if required) |
| n+2 | n+3 | 00 | 00 | Block size of 0000h - indicates end of the source program |

### *4.10.5 Boot Data Stream Structure*

This section details the data transfer protocols or stream structures that allow boot data transfer between boot ROM and host device. This data transfer protocol is compatible to the respective bootloaders on other C2000 devices.

#### 4.10.5.1 Bootloader Data Stream Structure

Table 4-28 and Example 4-1 show the structure of the data stream incoming to the bootloader. The basic structure is the same for all the bootloaders and is based on the C54x source data stream generated by the C54x hex utility. The C28x hex utility (hex2000.exe) has been updated to support this structure. The hex2000.exe utility is included with the C2000 code generation tools. All values in the data stream structure are in hex.

The first 16-bit word in the data stream is known as the key value. The key value is used to 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 and for a 16-bit 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 it. If a bootloader does not use these values then they are reserved for future use and the bootloader



Copyright © 2024 Texas Instruments Incorporated

simply reads the value and then discards it. Currently only the SPI and I2C and parallel bootloaders use these words to initialize registers.

The tenth and eleventh words comprise the 22-bit entry point address. This address is used to initialize the PC after the boot load is complete. This address is most likely the entry point of the program downloaded by the bootloader.

The twelfth word in the data stream is the size of the first data block to be transferred. The size of the block is defined as 8-bit data stream format. For example, to transfer a block of 20 8-bit data values from an 8-bit data stream, the block size 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 that cleans up and exits. Execution then continues at the entry point address as determined by the input data stream contents.

**Table 4-28. LSB/MSB Loading Sequence in 8-Bit Data Stream**

| Byte | | Contents | |
|---|---|---|---|
| | | LSB (First Byte of 2) | MSB (Second Byte of 2) |
| 1 | 2 | LSB: AA (KeyValue for memory width = 8 bits) | MSB: 08h (KeyValue for memory width = 8 bits) |
| 3 | 4 | LSB: Register initialization value or reserved | MSB: Register initialization value or reserved |
| 5 | 6 | LSB: Register initialization value or reserved | MSB: Register initialization value or reserved |
| 7 | 8 | LSB: Register initialization value or reserved | MSB: Register initialization value or reserved |
| ... | ... | ... | ... |
| 17 | 18 | LSB: Register initialization value or reserved | MSB: Register initialization value or reserved |
| 19 | 20 | LSB: Upper half of Entry point PC[23:16] | MSB: Upper half of entry point PC[31:24] (Always 0x00) |
| 21 | 22 | LSB: Lower half of Entry point PC[7:0] | MSB: Lower half of Entry point PC[15:8] |
| 23 | 24 | LSB: Block size in words of the first block to load. If the block size is 0, this indicates the end of the source program. Otherwise another block follows. For example, a block size of 0x000A 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 |



Copyright © 2024 Texas Instruments Incorporated

***Example 4-1. Data Stream Structure 8-bit***

```
AA 08          ; 0x08AA 8-bit key value
00 00 00 00    ; 8 reserved words
00 00 00 00
00 00 00 00
00 00 00 00
3F 00 00 80    ; 0x003F8000 EntryAddr, starting point after boot load completes
05 00          ; 0x0005 - First block consists of 5 16-bit words
3F 00 10 90    ; 0x003F9010 - First block will be loaded starting at 0x3F9010
01 00          ; Data loaded = 0x0001 0x0002 0x0003 0x0004 0x0005
02 00
03 00
04 00
05 00
02 00          ; 0x0002 - 2nd block consists of 2 16-bit words
3F 00 00 80    ; 0x003F8000 - 2nd block will be loaded starting at 0x3F8000
00 77          ; Data loaded = 0x7700 0x7625
25 76
00 00          ; 0x0000 - Size of 0 indicates end of data stream
After load has completed the following memory values will have been initialized as follows:
Location   Value
0x3F9010   0x0001
0x3F9011   0x0002
0x3F9012   0x0003
0x3F9013   0x0004
0x3F9014   0x0005
0x3F8000   0x7700
0x3F8001   0x7625
PC Begins execution at 0x3F8000
```

### *4.10.6 GPIO Assignments*

This section details the GPIOs and boot options used for each boot mode set in the BMODE bit-field of the BOOTCTRL register in OTP memory. Refer to Section 4.5 on the details of the BOOTCTRL fields.

**Note**

The peripheral bootloader GPIO configurations only apply to CPU1. CPU1 user application can configure any available first instance of a peripheral (SCIA, SPIA, CANA, and so forth) GPIO for CPU2 to use for a CPU2 particular bootloader.

**Table 4-29. SCI Boot Options**

| Option | BMODE Value | SCITXDA GPIO | SCIRXDA GPIO |
|---|---|---|---|
| 0 (default) | 0x01 | GPIO84 | GPIO85 |
| 1 | 0x81 | GPIO29 | GPIO28 |

**Table 4-30. CAN Boot Options**

| Option | BMODE Value | CANTXA GPIO | CANRXA GPIO |
|---|---|---|---|
| 0 | 0x07 | GPIO71 | GPIO70 |
| 1 | 0x87 | GPIO63 | GPIO62 |

**Table 4-31. I2C Boot Options**

| Option | BMODE Value | SDAA GPIO | SCLA GPIO |
|---|---|---|---|
| 0 | 0x05 | GPIO91 | GPIO92 |
| 1 | 0x85 | GPIO32 | GPIO33 |



Copyright © 2024 Texas Instruments Incorporated

**Table 4-32. USB Boot Options**

| Option | BMODE Value | USBDM GPIO | USBDP GPIO |
|---|---|---|---|
| 0 | 0x0C | GPIO42 | GPIO43 |

**Table 4-33. RAM Boot Options**

| Option | BMODE Value | RAM Entry Point (Address) | CPU Support |
|---|---|---|---|
| 0 | 0x0A | 0x0000 0000 | CPU1 and CPU2 |

**Table 4-34. Flash Boot Options**

| Option | BMODE Value | Flash Entry Point (Address) | Flash Sector | CPU Support |
|---|---|---|---|---|
| 0 | 0x0B | 0x0008 0000 | Sector A | CPU1 and CPU2 |

**Table 4-35. Wait Boot Options**

| Option | BMODE Value | CPU Support |
|---|---|---|
| 0 (default) | 0x02 | CPU1 and CPU2 |

**Table 4-36. SPI Boot Options**

| Option | BMODE Value | SPISIMOA | SPISOMIA | SPICLKA | SPISTEA |
|---|---|---|---|---|---|
| 0 | 0x04 | GPIO58 | GPIO59 | GPIO60 | GPIO61 |
| 1 | 0x84 | GPIO16 | GPIO17 | GPIO18 | GPIO19 |

**Table 4-37. Parallel Boot Options**

| Option | BMODE Value | D0-D7 GPIO | DSP Control GPIO | Host Control GPIO | CPU Support |
|---|---|---|---|---|---|
| 0 (default) | 0x0 | D0 - GPIO65 | GPIO69 | GPIO70 | CPU1 and CPU2 |
| | | D1 - GPIO64 | | | |
| | | D2 - GPIO58 | | | |
| | | D3 - GPIO59 | | | |
| | | D4 - GPIO60 | | | |
| | | D5 - GPIO61 | | | |
| | | D6 - GPIO62 | | | |
| | | D7 - GPIO63 | | | |



Copyright © 2024 Texas Instruments Incorporated

### 4.10.7 Secure ROM Function APIs

Within secure ROM of each core, functions are available to be called by the application to perform EXEONLY Flash/RAM tasks in a secure manner.

**Note**

The application must disable interrupts before calling one of the EXEONLY function APIs.

If a vector fetch request is given by the CPU while the program counter (PC) is within the EXEONLY function API code of the Secure ROM, a reset fires. The consequence of this is if an NMI or ITRAP occurs while the PC is executing one of the EXEONLY API functions, the NMI/ITRAP cannot be serviced because a reset is fired to that subsystem.

The **safe copy code zone 1 and zone 2 functions** 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 4-38. Safe Copy Code Function**

| CPU | Function Prototype | Function Parameters | Function Return Value |
|---|---|---|---|
| CPU1, CPU2 | Uint16 **SafeCopyCodeZ1**(Uint32 size, Uint16 *dst, Uint16 *src)<br><br>Uint16 **SafeCopyCodeZ2**(Uint32 size, Uint16 *dst, Uint16 *src) | *size* : The number of 16-bit words to copy<br><br>*dst* : The destination memory address in EXEONLY RAM<br><br>*src* : The source memory address in EXEONLY Flash | 0xXXXX : Returns the number of 16-bit words copied<br><br>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/or RAM are not set to EXEONLY; Error occurred during data copy |

The **safe CRC calculation zone 1 and zone 2 functions** 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 CRC'd is returned.

**Table 4-39. Safe CRC Calculation Function**

| CPU | Function Prototype | Function Parameters | Function Return Value |
|---|---|---|---|
| CPU1, CPU2 | Uint16 **SafeCRCCalcZ1**(Uint16 len_id, Uint16 *dst, Uint16 *src)<br><br>Uint16 **SafeCRCCalcZ2**(Uint16 size, Uint16 *dst, Uint16 *src) | *len_id* : A number from 1 to 8 that corresponds to length options of 32, 64, 128, 256, 512, 1024, 2048, or 4096 16-bit words<br><br>*dst* : The destination memory address for resulting CRC<br><br>*src* : The source memory address to begin CRC calculation | 0xXXXX : Returns the number of 16-bit words CRC'd<br><br>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 |



Copyright © 2024 Texas Instruments Incorporated

### 4.10.8 Boot IPC

For CPU1 and CPU2 to communicate during the boot process, a set of inter-processor communication (IPC) registers are used. Each core polls for supported IPC commands that determine what specific action should be performed. Actions include performing reads or writes to a particular memory address, branching to an address, or calling a function. The supported commands differ depending on the core.

Prior to sending an IPC command to CPU2 boot ROM, the CPU1 application should perform the necessary GPIO mux configurations for the peripheral IO pins. CPU2 peripheral loaders do not configure any of the GPIO mux options and only configure the peripheral as required for the application load. Once configured, CPU1 can then assign the peripheral to CPU2 through configuration of the CPU select register.

#### 4.10.8.1 CPU1 IPC Commands

This section details the commands CPU1 boot ROM supports. These commands can be used by CPU2 applications to have CPU1 boot ROM to perform an action such as configure peripherals or IOs. To use these IPC commands, CPU1 boot ROM must be in Wait Boot mode. Additionally, watchdog is disabled when using IPC.

**Note**

All 32-bit operations are done in little-endian format (C28x is 16-bit addressable). Example: a 32-bit IPC write is handled as:

- Data[15:0] is written in address
- Data [31:16] is written in address + 1

**Table 4-40. C2TOC1IPC Commands Table**

| Value | IPCRECVCOM (CPU2 - R/W, CPU1- R) | IPCRECVADDR (CPU2 - R/W, CPU1- R) | IPCRECVDATA (CPU2 - R/W, CPU1- R) | IPCLOCALREPLY (CPU2 - R, CPU1 - R/W) | C2TOC1IPCFLG [31] = ? C2TOC1IPCFLAG [0] = 0 | Description |
|---|---|---|---|---|---|---|
| 0 | C2C1_BROM_IPC_ COMMAND_ILLEGAL | NOT-USED | NOT-USED | Look at error codes table | 0x01 | Illegal command |
| 1 | C2C1_BROM_IPC_SET_ BITS_16 | Address of the 16-bit register | Data in C2TOC1IPCDAT AW[15:0] | Data read back from address after write | 0x00 = Command success | *(address) \|= data; |
| 2 | C2C1_BROM_IPC_SET_ BITS_32 | Address of the 32-bit register | Data; | Data read back from address after write | Same as above | *(address) \|= data; |
| 3 | C2C1_BROM_IPC_CLEAR_ BITS_16 | Address of the 16-bit register | Data in C2TOC1IPCDAT AW[15:0] | Data read back after write | Same as above | *(address) &= ~data; |
| 4 | C2C1_BROM_IPC_CLEAR_ BITS_32 | Address of the 32-bit register | Data | Data read back after write | Same as above | *(address) &= ~data; |
| 5 | C2C1_BROM_IPC_DATA_ WRITE_16 | Address of the 16-bit register | Data in MTOCIPCDATA W[15:0] | Data read back from the address | Same as above | *(address) = data; |
| 6 | C2C1_BROM_IPC_DATA_ WRITE_32 | Address of the 32-bit register | Data | Same as above | Same as above | *(address) = data; |
| 7 | C2C1_BROM_IPC_DATA_ READ_16 | Address of the 16-bit register | --NOT_USED-- | Data in C2TOC1IPCDATA R[15:0] | Same as above | C2TOC1IPCDATA R[15:0] = *(address); Only 16 bit read from address |
| 8 | C2C1_BROM_IPC_DATA_ READ_32 | Address of the 32-bit register | Same as above | 32-bit data | Same as above | C2TOC1IPCDATA R[31:0] = *(address); 32 bits read from address |



Copyright © 2024 Texas Instruments Incorporated

**Table 4-40. C2TOC1IPC Commands Table (continued)**

| Value | IPCRECVCOM (CPU2 - R/W, CPU1- R) | IPCRECVADDR (CPU2 - R/W, CPU1- R) | IPCRECVDATA (CPU2 - R/W, CPU1- R) | IPCLOCALREPLY (CPU2 - R, CPU1 - R/W) | C2TOC1IPCFLG [31] = ? C2TOC1IPCFLAG [0] = 0 | Description |
|---|---|---|---|---|---|---|
| 9 | C2C1_BROM_IPC_SET_ BITS_ PROTECTED_16 | Address of the 16-bit register | Data in C2TOC1IPCDATAW[15:0] | Data read back from address after write | 0x00 = Command success | EALLOW; *(address) \|= data; EDIS; |
| 10 | C2C1_BROM_IPC_SET_ BITS_ PROTECTED_32 | Address of the 32-bit register | Data; | Data read back from address after write | Same as above | EALLOW; *(address) \|= data; EDIS; |
| 11 | C2C1_BROM_IPC_CLEAR_ BITS_ PROTECTED_16 | Address of the 16-bit register | Data in C2TOC1IPCDATAW[15:0] | Data read back after write | Same as above | EALLOW; *(address) &= ~data; EDIS; |
| 12 | C2C1_BROM_IPC_CLEAR_ BITS_ PROTECTED_32 | Address of the 32-bit register | Data | Data read back after write | Same as above | EALLOW; *(address) &= ~data; EDIS; |
| 13 | C2C1_BROM_IPC_DATA_ WRITE_ PROTECTED_16 | Address of the 16-bit register | Data in C2TOC1IPCDATAW[15:0] | Data read back from the address | Same as above | EALLOW; *(address) &= ~data; EDIS; |
| 14 | C2C1_BROM_IPC_DATA_ WRITE_ PROTECTED_32 | Address of the 32-bit register | Data | Same as above | Same as above | EALLOW; *(address) &= ~data; EDIS; |
| 15 | C2C1_BROM_IPC_DATA_ READ_ PROTECTED_16 | Address of the 32-bit register | --NOT_USED-- | Data in C2TOC1IPCDATAR[15:0] | Same as above | EALLOW; C2TOC1IPCDATAR[15:0] = *(address); EDIS; Only 16 bit read from address |
| 16 | C2C1_BROM_IPC_DATA_ READ_ PROTECTED_32 | Address of the 32-bit register | Data | Same as above | Same as above | EALLOW; C2TOC1IPCDATAR[31:0] = *(address); EDIS; 32 bits read from address |



Copyright © 2024 Texas Instruments Incorporated

#### 4.10.8.2 CPU2 IPC Commands

This section details the commands that CPU2 boot ROM supports. These commands can be used by CPU1 applications to have CPU2 perform a specific action. To use these IPC commands, CPU2 boot ROM must be in Wait Boot mode.

**Table 4-41. C1TOC2IPC Commands Table**

| Value | IPCRECVCOM (CPU1 - R/W, CPU2 R) | IPCRECVADDR (CPU1 - R/W, BCPU2- R) | IPCRECVDATA (CPU1 - R/W, CPU2- R) | IPCLOCALREPLY (CPU1 - R, CPU2 - R/W) | C1TOC2IPCFLG[31] = ? C1TOC2IPCFLAG[0] = 0 | Description |
|---|---|---|---|---|---|---|
| 0 | C1C2_BROM_IPC_ COMMAND_ILLEGAL | NOT-USED | NOT-USED | Look at error codes table | 0x01 | Illegal command |
| 1 | C1C2_BROM_IPC_SET_ BITS_16 | Address of the 16-bit register | Data in C1TOC2IPCDATA W[15:0] | Data read back from address after write | 0x00 = Command success | *(address) \|= data; |
| 2 | C1C2_BROM_IPC_SET_ BITS_32 | Address of the 32-bit register | Data; | Data read back from address after write | Same as above | *(address) \|= data; |
| 3 | C1C2_BROM_IPC_CLEAR_ BITS_16 | Address of the 16-bit register | Data in C1TOC2IPCDATA W[15:0] | Data read back after write | Same as above | *(address) &= ~data; |
| 4 | C1C2_BROM_IPC_CLEAR_ BITS_32 | Address of the 32-bit register | Data | Data read back after write | Same as above | *(address) &= ~data; |
| 5 | C1C2_BROM_IPC_DATA_ WRITE_16 | Address of the 16-bit register | Data in MTOCIPCDATA W[15:0] | Data read back from the address | Same as above | *(address) = data; |
| 6 | C1C2_BROM_IPC_DATA_ WRITE_32 | Address of the 32-bit register | Data | Same as above | Same as above | *(address) = data; |
| 7 | C1C2_BROM_IPC_DATA_ READ_16 | Address of the 16-bit register | NOT-USED | Data in C1TOC2IPCDATA R[15:0] | Same as above | C1TOC2IPCDATA R[15:0] = *(address); 16 bits read from address |
| 8 | C1C2_BROM_IPC_DATA_ READ_32 | Address of the 32-bit register | Same as above | 32 bit data | Same as above | C1TOC2IPCDATA R[31:0] = *(address); 32 bits read from address |
| 9 | C1C2_BROM_IPC_SET_ BITS_ PROTECTED_16 | Address of the 16-bit register | Data in C1TOC2IPCDATA W[15:0] | Data read back from address after write | 0x00 = Command success | EALLOW; *(address) \|= data; EDIS; |
| 10 | C1C2_BROM_IPC_SET_ BITS_ PROTECTED_32 | Address of the 32-bit register | Data; | Data read back from address after write | Same as above | EALLOW; *(address) \|= data; EDIS; |
| 11 | C1C2_BROM_IPC_CLEAR_ BITS_ PROTECTED_16 | Address of the 16-bit register | Data in C1TOC2IPCDATA W[15:0] | Data read back after write | Same as above | EALLOW; *(address) &= ~data; EDIS; |
| 12 | C1C2_BROM_IPC_CLEAR_ BITS_ PROTECTED_32 | Address of the 32-bit register | Data | Data read back after write | Same as above | EALLOW; *(address) &= ~data; EDIS; |
| 13 | C1C2_BROM_IPC_DATA_ WRITE_ PROTECTED_16 | Address of the 16-bit register | Data in C1TOC2IPCDATA W[15:0] | Data read back from the address | Same as above | EALLOW; *(address) = data; EDIS; |
| 14 | C1C2_BROM_IPC_DATA_ WRITE_ PROTECTED_32 | Address of the 32-bit register | Data | Same as above | Same as above | EALLOW; *(address) = data; EDIS; |



Copyright © 2024 Texas Instruments Incorporated

**Table 4-41. C1TOC2IPC Commands Table (continued)**

| Value | IPCRECVCOM (CPU1 - R/W, CPU2 R) | IPCRECVADDR (CPU1 - R/W, BCPU2- R) | IPCRECVDATA (CPU1 - R/W, CPU2- R) | IPCLOCALREPLY (CPU1 - R, CPU2 - R/W) | C1TOC2IPCFLG[31] = ? C1TOC2IPCFLAG[0] = 0 | Description |
|---|---|---|---|---|---|---|
| 15 | C1C2_BROM_IPC_DATA_READ_ PROTECTED_16 | Address of the 16-bit register | NOT-USED | Data in C1TOC2IPCDATAR[15:0] | Same as above | EALLOW; C1TOC2IPCDATAR[15:0] = *(address); 16 bits read from address |
| 16 | C1C2_BROM_IPC_DATA_READ_ PROTECTED_32 | Address of the 32-bit register | Same as above | 32 bit data | Same as above | EALLOW; C1TOC2IPCDATAR[31:0] = *(address); EDIS; 32 bits read from address |
| 17 | C1C2_BROM_IPC_ BRANCH_CALL | Address where to branch to | _NOT-USED_BY_BOOTROM (code at the branch can use this though) | NOT-USED_BY_BOOTROM (code at the branch can use this though) | Same as above | C2-BootROM jumps to the address in ADDR register and starts executing the code from that address. PIE is enabled when this branch occurs, it is up to the application to disable and reload PIE interrupt handlers if it wants to. |
| 18 | C1C2_BROM_IPC_ FUNCTION_CALL | Address of the function | Parameter for the function call | Result of function call (return value if any from function) | Same as above | C2-BootROM jumps to the address in ADDR register and starts executing the code from that address; Data in DATAW register can be used as parameter to the function call. C-BootROM returns back to where it was after servicing the function call. Function call is performed from inside the interrupt service routine on Aria so user has to keep this in mind. |



Copyright © 2024 Texas Instruments Incorporated

**Table 4-41. C1TOC2IPC Commands Table (continued)**

| Value | IPCRECVCOM (CPU1 - R/W, CPU2 R) | IPCRECVADDR (CPU1 - R/W, BCPU2- R) | IPCRECVDATA (CPU1 - R/W, CPU2- R) | IPCLOCALREPLY (CPU1 - R, CPU2 - R/W) | C1TOC2IPCFLG[31] = ? C1TOC2IPCFLAG[0] = 0 | Description |
|---|---|---|---|---|---|---|
| 19 | C1C2_BROM_IPC_ EXECUTE_ BOOTMODE_CMD<br>• C1TOC2BOOTMODE = 0xA, C1C2_BROM_ BOOTMODE_ BOOT_FROM_RAM<br>• C1TOC2BOOTMODE = 0xB, C1C2_BROM_ BOOTMODE_ BOOT_FROM_FLASH<br>• C1TOC2BOOTMODE = 0x1, C1C2_BROM_ BOOTMODE_ BOOT_FROM_SCI<br>• C1TOC2BOOTMODE = 0x4, C1C2_BROM_ BOOTMODE_ BOOT_FROM_SPI<br>• C1TOC2BOOTMODE = 0x5, C1C2_BROM_ BOOTMODE_ BOOT_FROM_I2C<br>• C1TOC2BOOTMODE = 0x0, C1C2_BROM_ BOOTMODE_ BOOT_FROM_ PARALLEL<br>• C1TOC2BOOTMODE = 0x7, C1C2_BROM_v BOOTMODE_ BOOT_FROM_CAN | NOT-USED | NOT-USED | NOT-USED | Same as above | Execute loaders as per requested value in C1TOC2BOOT MODE register. |



Copyright © 2024 Texas Instruments Incorporated

#### 4.10.8.3 CPU2 IPC Error Commands

This section details the commands CPU2 boot ROM supports. These commands can be used by CPU1 applications to have CPU2 perform a specific action. To use these IPC commands, CPU2 boot ROM must be in Wait Boot mode.

**Table 4-42. CPU2 Error Command Values**

| Error Value | Description | Note |
|---|---|---|
| 0x0000 0000 | Invalid Command Value | Default value when starting boot |
| 0xFFFF FFFE | CPU2 has got an ITRAP | Address where ITRAP occurred is placed in IPCADDR register |
| 0xFFFF FFFD | CPU2 got a spurious PIE interrupt | Interrupt number placed in IPCDATAW register |
| 0xFFFF FFFC | CPU2 got a PIE vector mismatch error | - |
| 0xFFFF FFFB | CPU2 got an uncorrectable Flash error | - |
| 0xFFFF FFFA | CPU2 got an uncorrectable RAM error | - |

### *4.10.9 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, XRS, and HIBERNATE reset types. For all other resets, the boot ROM starts executing with the clocks that were already set up before reset.

Only CPU1 performs the clock configuration for the device during boot up.

**Table 4-43. 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. |

**Table 4-44. Clock State After Boot ROM**

| Reset Source | Clock State |
|---|---|
| POR/XRS/HIBERNATE | Bypassed PLL. |
| | PLL multiplier is set to 0x0 |
| | Clock divider is set to /1. |
| All other Resets | Maintain clocks setup before device reset. |

**Note**

If the PLL is used during the boot process, the PLL is bypassed by the boot ROM code before branching to the user application.



Copyright © 2024 Texas Instruments Incorporated

### 4.10.10 Wait State Configuration

This section details the ROM memory wait state configurations. By default, the CPU1 and CPU2 ROM memory on this device is not zero-wait state; the ROM memory is 1-wait state with prefetch disabled. ROM does support prefetch enable and disable configurations to provide better execution speeds at varying clock frequencies. Configuring the wait state enables user applications to adjust for when performing callbacks into ROM or secure copy code (SCC).

**Table 4-45. ROM Wait States**

| Wait State Disable Bit ROMWAITSTATE Register (Bit 0 – 0x5F540) | Pre-Fetch Enable Bit ROMPREFETCH Register (Bit 0 – 0x5E608) | C28x ROM Configuration |
|---|---|---|
| 0 | 0 | • Wait state enabled<br>• Prefetch disabled<br>• Maximum Frequency: 200MHz |
| 0 | 1 | • Wait state enabled<br>• Prefetch enabled<br>• Maximum Frequency: 180MHz |
| 1 | Don't Care | • 0 Wait state<br>• Prefetch disabled<br>• Maximum Frequency: 150MHz |

### 4.10.11 Boot Status information

Boot ROM keeps a record of the different events that can occur during boot ROM execution. This is because NMI and other exceptions are enabled by default in the device, and must be handled accordingly. Boot ROM stores the boot status information in a RAM location so that the user application can look at this boot status and take the necessary actions per the application's needs to handle these events.

#### 4.10.11.1 CPU1 Booting Status

This section details the boot status RAM location and the bit field definitions for CPU1. When the specific bit field is set, the described event or action has occurred.

**Table 4-46. CPU1 Boot Status Address**

| Description | Address |
|---|---|
| CPU1 Boot ROM Status | 0x0000 002C |

**Table 4-47. CPU1 Boot Status Bit Fields**

| Bit | Description |
|---|---|
| 31 | CPU1 Boot ROM has finished running |
| 30 | Boot ROM detected a missing clock NMI |
| 29 | Boot ROM detected a RAM bit error NMI |
| 28 | Boot ROM detected a Flash bit error NMI |
| 27 | Boot ROM detected CPU1 HWBIST error NMI |
| 26 | Boot ROM detected CPU2 HWBIST error NMI |
| 25 | Boot ROM detected PIE vector error NMI |
| 22 | Boot ROM detected CPU2 watchdog reset |
| 21 | Boot ROM detected CPU2 NMI watchdog reset |
| 20 | Boot ROM detected OVF NMI |
| 19 | Boot ROM detected a PIE mismatch |
| 18 | Boot ROM detected CPU1 to CPU2 branch |



Copyright © 2024 Texas Instruments Incorporated

**Table 4-47. CPU1 Boot Status Bit Fields (continued)**

| Bit | Description |
|---|---|
| 17 | Boot ROM detected an ITRAP |
| 15 | Boot ROM handled POR |
| 14 | Boot ROM handled XRS |
| 13 | Boot ROM handled HWBIST reset |
| 12 | Boot ROM handled hibernate reset |
| 11 | Boot ROM handled all the resets |
| 10 | DCSM initialization has completed |
| 9 | Flash boot has started |
| 8 | CPU1 Boot ROM has started running |

#### 4.10.11.2 CPU1 Boot Mode Status

This section details the boot mode that is determined during CPU1 boot.

**Table 4-48. CPU1 Boot Mode Status Address**

| Description | Address |
|---|---|
| CPU1 Boot Mode Address | 0x0000 0038 |

**Table 4-49. CPU1 Boot Mode Status Values**

| Boot Mode | Value |
|---|---|
| Parallel Boot | 0x0 |
| SCI Boot | 0x1 |
| SCI Boot (ALT) | 0x81 |
| Wait Boot | 0x2 |
| Get Boot | 0x3 |
| SPI Boot | 0x4 |
| SPI Boot (ALT) | 0x84 |
| I2C Boot | 0x5 |
| I2C Boot (ALT) | 0x85 |
| CAN Boot | 0x7 |
| CAN Boot (ALT) | 0x87 |
| RAM Boot | 0xA |
| Flash Boot | 0xB |
| USB Boot | 0xC |
| USB Boot (ALT) | 0x8C |



Copyright © 2024 Texas Instruments Incorporated

#### 4.10.11.3 CPU2 Booting Status

This section details the boot status bit field definitions for CPU2. When the specific bit field is set, the described event or action has occurred. These status bits can be read from the C2TOC1BOOTSTS register.

**Table 4-50. CPU2 Boot Status Address**

| Description | Address |
|---|---|
| CPU2 Boot ROM Status | 0x0000 0002 |

**Table 4-51. CPU2 Boot Status Bit Fields**

| Bit | Description |
|---|---|
| 31 | CPU2 Boot ROM has finished running |
| 30 | Boot ROM detected a missing clock NMI |
| 29 | Boot ROM detected a RAM bit error NMI |
| 28 | Boot ROM detected a Flash bit error NMI |
| 27 | Boot ROM detected CPU1 HWBIST error NMI |
| 26 | Boot ROM detected CPU2 HWBIST error NMI |
| 25 | Boot ROM detected PIE vector error NMI |
| 22 | Boot ROM detected CPU2 watchdog reset |
| 21 | Boot ROM detected CPU2 NMI watchdog reset |
| 20 | Boot ROM detected OVF NMI |
| 19 | Boot ROM detected a PIE mismatch |
| 18 | Boot ROM detected CPU1 to CPU2 branch |
| 17 | Boot ROM detected an ITRAP |
| 11 | Boot ROM handled all the resets |
| 10 | DCSM initialization has completed |
| 9 | Flash boot has started |
| 8 | CPU2 Boot ROM has started running |
| 3-0 | 0 – Invalid Status. CPU2 hasn't set a valid status yet<br>1 – CPU2 Boot ROM has started running<br>2 – CPU2 Boot ROM has completed and is ready for IPC commands<br>3 – CPU2 ACKs the boot command in C1TOC2BOOTMODE register<br>4 – CPU2 doesn't support the command in C1TOC2BOOTMODE register<br>5 – CPU2 NAKs the boot command in C1TOC2BOOTMODE register |

#### 4.10.11.4 CPU1 IPC NAK Status

Table 4-52 details the NAK status information for CPU1, if boot ROM IPC command support is enabled. These NAK status bits are returned by CPU1 in the C2TOC1IPCDATAR[11:0] register.

**Table 4-52. CPU1 IPC NAK Status Bit Fields**

| Bit | Description |
|---|---|
| 3-0 | 0 – Invalid Value or value isn't set<br>1 – Command not supported<br>2 – Command not set correctly<br>3 – CPU is trying to send a second command before the first one is complete<br>4 – Command execution resulted in an error<br>5 – Command cannot be executed in the current state of the boot ROM |



Copyright © 2024 Texas Instruments Incorporated

#### 4.10.11.5 CPU2 IPC NAK Status

Table 4-53 details the NAK status information for CPU2, if boot ROM IPC command support is enabled. These NAK status bits are returned by CPU1 in the C1TOC2IPCDATAR[11:0] register.

**Table 4-53. CPU2 IPC NAK Status Bit Fields**

| Bit | Description |
|---|---|
| 3-0 | 0 – Invalid Value or value isn't set<br>1 – Command not supported<br>2 – Command not set correctly<br>3 – CPU1 is trying to send a second command before the first one is complete<br>4 – Command execution resulted in an error<br>5 – Command cannot be executed in the current state of the boot ROM |

### *4.10.12 ROM Version*

The ROM revision and release date information is stored at the ROM locations specified in Table 4-54.

**Table 4-54. Boot ROM Version Information for CPU1 and CPU2**

| Start Address | End Address | Contents |
|---|---|---|
| 0x003F FF7A | 0x003F FF7A | Revision Number |
| 0x003F FF7B | 0x003F FF7B | Revision Date |

Interpreting the contents:

- Reading a revision value of 0x100 represents version 1.0.
- Reading a revision date value of 0x0715 represents 07/15 or July 2015.



Copyright © 2024 Texas Instruments Incorporated

Chapter 5

# Direct Memory Access (DMA)

The direct memory access (DMA) module provides a hardware method of transferring data between peripherals and memory without intervention from the CPU; thereby, freeing up bandwidth for other system functions. Additionally, the DMA has the capability to orthogonally rearrange the data as the data is transferred as well as “ping-pong” data between buffers. These features are useful for structuring data into blocks for CPU processing.


**5.1 Introduction** ..... 664
**5.2 Architecture** ..... 666
**5.3 Address Pointer and Transfer Control** ..... 671
**5.4 Pipeline Timing and Throughput** ..... 677
**5.5 CPU and CLA Arbitration** ..... 678
**5.6 Channel Priority** ..... 679
**5.7 Overrun Detection Feature** ..... 680
**5.8 Software** ..... 681
**5.9 DMA Registers** ..... 682





Copyright © 2024 Texas Instruments Incorporated

## 5.1 Introduction

The strength of a controller is not measured purely in processor speed, but in total system capabilities. As a part of the equation, any time the CPU bandwidth for a given function can be reduced, the greater the system capabilities. Many times applications spend a significant amount of the bandwidth moving data, whether moving data from off-chip memory to on-chip memory, from a peripheral such as an analog-to-digital converter (ADC) to RAM, or from one peripheral to another. Furthermore, many times this data comes in a format that is not conducive to the optimum processing powers of the CPU. The DMA module described in this chapter has the ability to free up CPU bandwidth and rearrange the data into a pattern for more streamlined processing.

The DMA module is an event-based machine, meaning the DMA module requires a peripheral or software trigger to start a DMA transfer. Although the DMA module can be made into a periodic time-driven machine by configuring a timer as the DMA trigger source, there is no mechanism within the module to start memory transfers periodically. The DMA module has six independent DMA channels that can be configured separately and each channel contains an independent PIE interrupt to let the CPU know when a DMA transfer has either started or completed. Five of the six channels are exactly the same, while Channel 1 has the ability to be configured at a higher priority than the others. At the heart of the DMA is a state machine and tightly coupled address control logic. This address control logic allows for rearrangement of the block of data during the transfer as well as the process of ping-ponging data between buffers. Each of these features is discussed in detail in this chapter.

### *5.1.1 Features*

DMA features include:

- Six channels with independent PIE interrupts
- Each DMA channel can be triggered from multiple peripheral trigger sources independently
- Word Size: 16-bit or 32-bit (SPI limited to 16-bit)
- Throughput: 3 cycles/word without arbitration



Copyright © 2024 Texas Instruments Incorporated

### 5.1.2 Block Diagram

Figure 5-1 shows a device-level block diagram of the DMA.

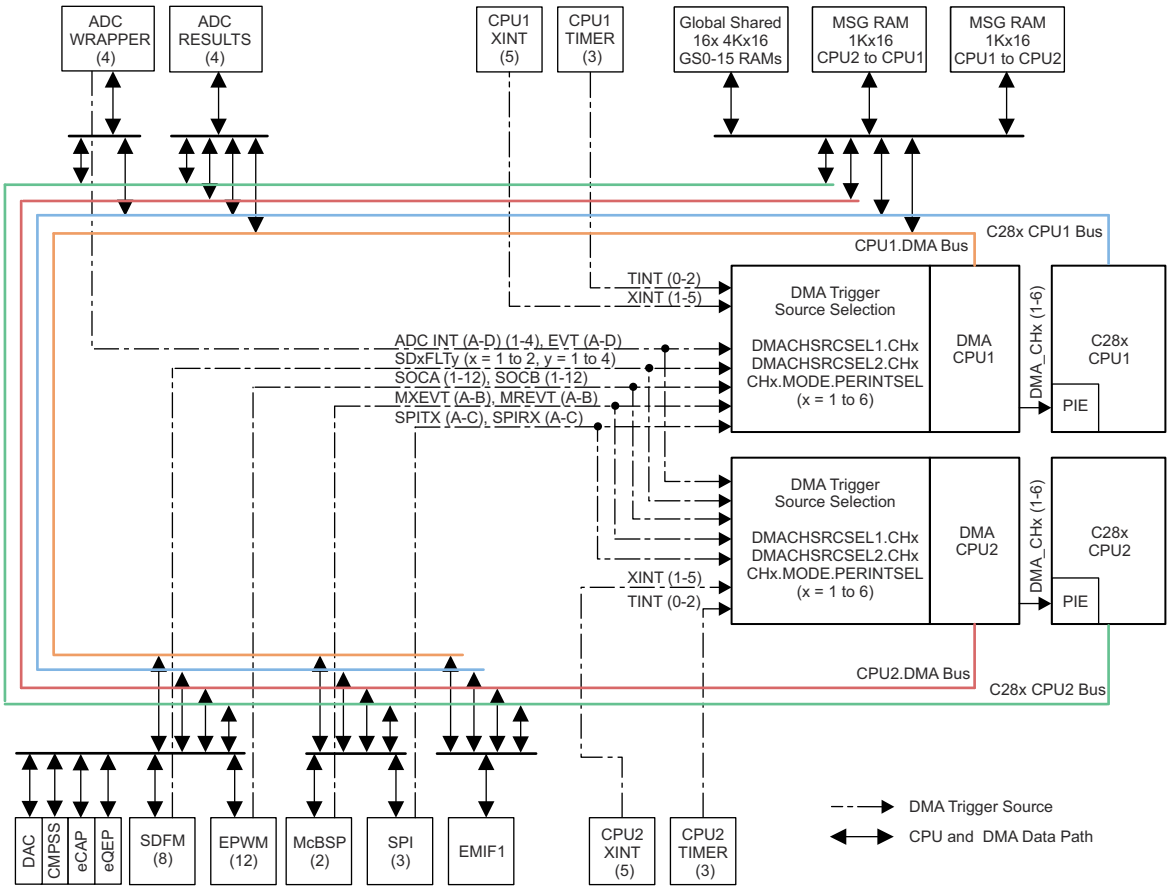



**Figure 5-1. DMA Block Diagram**



Copyright © 2024 Texas Instruments Incorporated

## 5.2 Architecture

### 5.2.1 Common Peripheral Architecture

There are two CPU subsystems; the CPU1 subsystem and CPU2 subsystem, with each containing a CLA and a DMA. The architecture allows several peripherals to be common between the two subsystems. Based on application need, these common peripherals can be attached to one of the two subsystems. Figure 5-2 shows how the CPUs and subsystems can be connected to the peripherals on peripheral frames 1 and 2. The clock, clock-enable, and reset muxing for the common peripherals are described in detail in other sections of this document.

Refer to Section 5.5 for more details on the arbitration scheme for all masters.

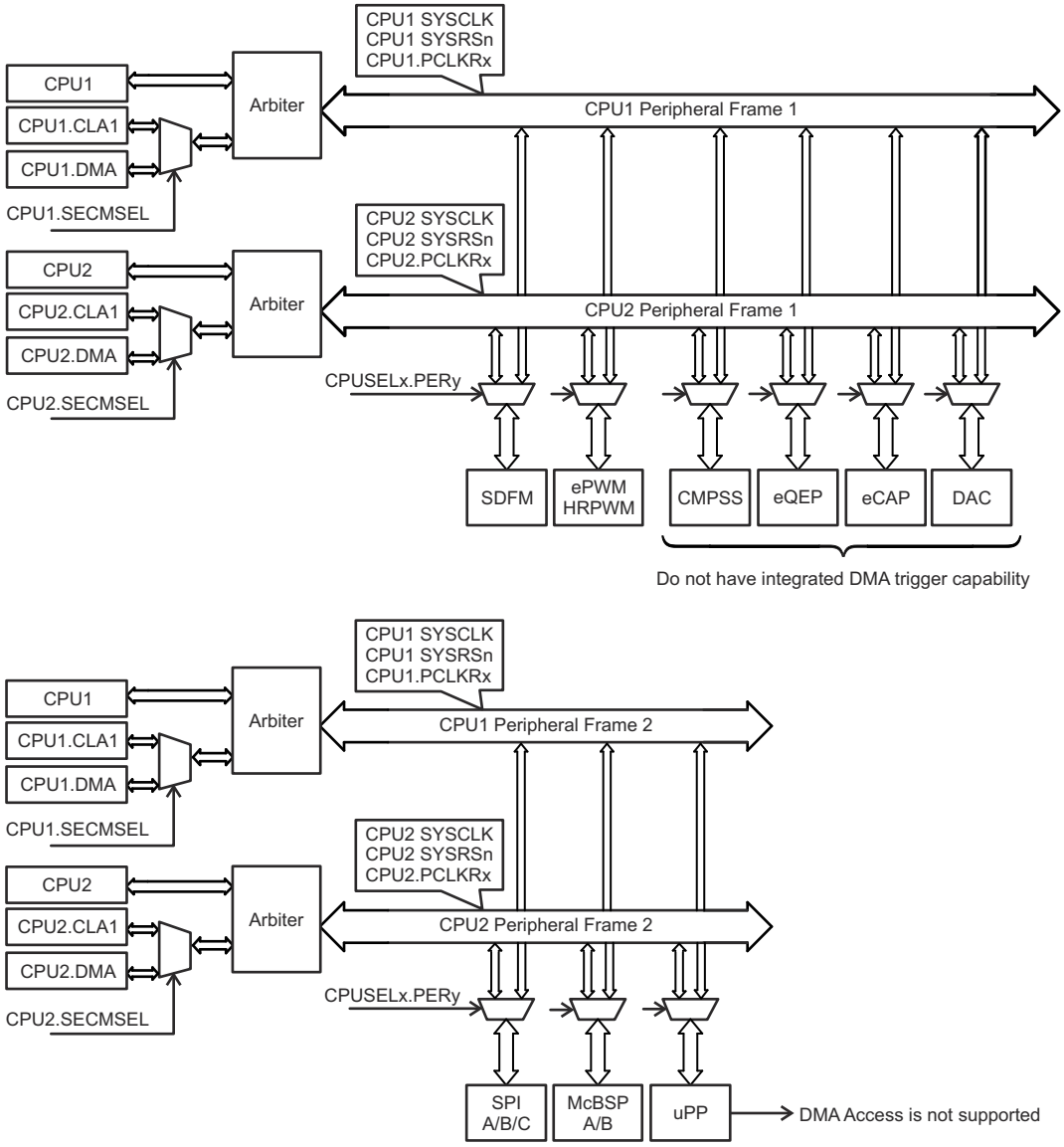



**Figure 5-2. Common Peripheral Architecture**

**Note**

If CPU and DMA make an access to the same peripheral frame in the same cycle, the DMA has priority and the CPU is stalled.



Copyright © 2024 Texas Instruments Incorporated

A CPUSEL bit associated with each peripheral defines whether the peripheral belongs to the CPU1 or CPU2 subsystem. If a peripheral belongs to a CPU subsystem, the peripheral can be accessed by the CPU and one of the secondary masters (DMA or CLA1). Refer to CPUSELx register definition for more details. The secondary master is statically selected using the SECMSEL register mapped to the respective CPU. Refer to CPUx.SECMSEL register definition for more details. If a secondary master is not selected, all writes from that master are ignored and all reads return 0x0 to any of the peripherals.

Similarly, if a peripheral does not belong to a CPU subsystem (as defined by the associated CPUSEL bit), all writes to that peripheral are ignored and all reads to that peripheral return 0x0 from any of the masters belonging to the unselected CPU subsystem. Note that since the arbiter has no knowledge regarding the ownership of individual peripherals (as can be seen from Figure 5-2), arbitration still happens even if the C28x or the selected secondary master tries to access a peripheral that does not belong to the CPU subsystem. See Section 5.5 for more information.

### *5.2.2 Peripheral Interrupt Event Trigger Sources*

Each DMA Channel can be configured to trigger by software and other peripheral triggers events. DMACHSRCSELx register can be used to configure DMA Trigger sources for each DMA channel. CHx.MODE.PERINTSEL register bit field can be set to channel number (CHx.MODE.PERINTSEL = x) as shown in Figure 5-3. Included in these DMA Trigger sources are five external interrupt signals that can be connected to most of the general-purpose input/output (GPIO) pins on the device. This adds significant flexibility to the event trigger capabilities. Upon receipt of a peripheral interrupt event signal, the DMA automatically sends a clear signal to the interrupt source so that subsequent interrupt events occur.

**Note**

To use the system-level DMA Trigger source selection, the DMA internal trigger source selection configuration for each channel can be done using the DMACHSRCSELx register and the CHx.MODE.PERINTSEL register. See Table 5-1 or the DMACHSRCSELx register definition for a complete list of DMA trigger sources.

Regardless of the value of the MODE.CHx[PERINTSEL] bit field, software can always force a trigger by using the CONTROL.CHx[PERINTFRC] bit. Likewise, software can always clear a pending DMA trigger using the CONTROL.CHx[PERINTCLR] bit.

Once a particular peripheral trigger event sets a channel's PERINTFLG bit, the bit remains pending until the priority logic of the state machine starts the burst transfer for that channel. Once the burst transfer starts, the flag is cleared. If a new peripheral trigger event is generated while a burst is in progress, the burst completes before responding to the new peripheral trigger event (after proper prioritization). If a third peripheral trigger event occurs before the pending event is serviced, an error flag is set in the CONTROL.CHx[OVRFLG] bit. If a peripheral trigger event occurs at the same time as the latched flag is being cleared, the trigger event has priority and the PERINTFLG remains set.

Figure 5-4 shows a diagram of the trigger select circuit.

Table 5-1 shows the peripheral trigger source options that are available for each channel.

**CAUTION**

See the Device Errata "ADC:DMA Read of Stale Result" Advisory regarding the potential for the DMA to read the ADC result registers before the result is ready



Copyright © 2024 Texas Instruments Incorporated

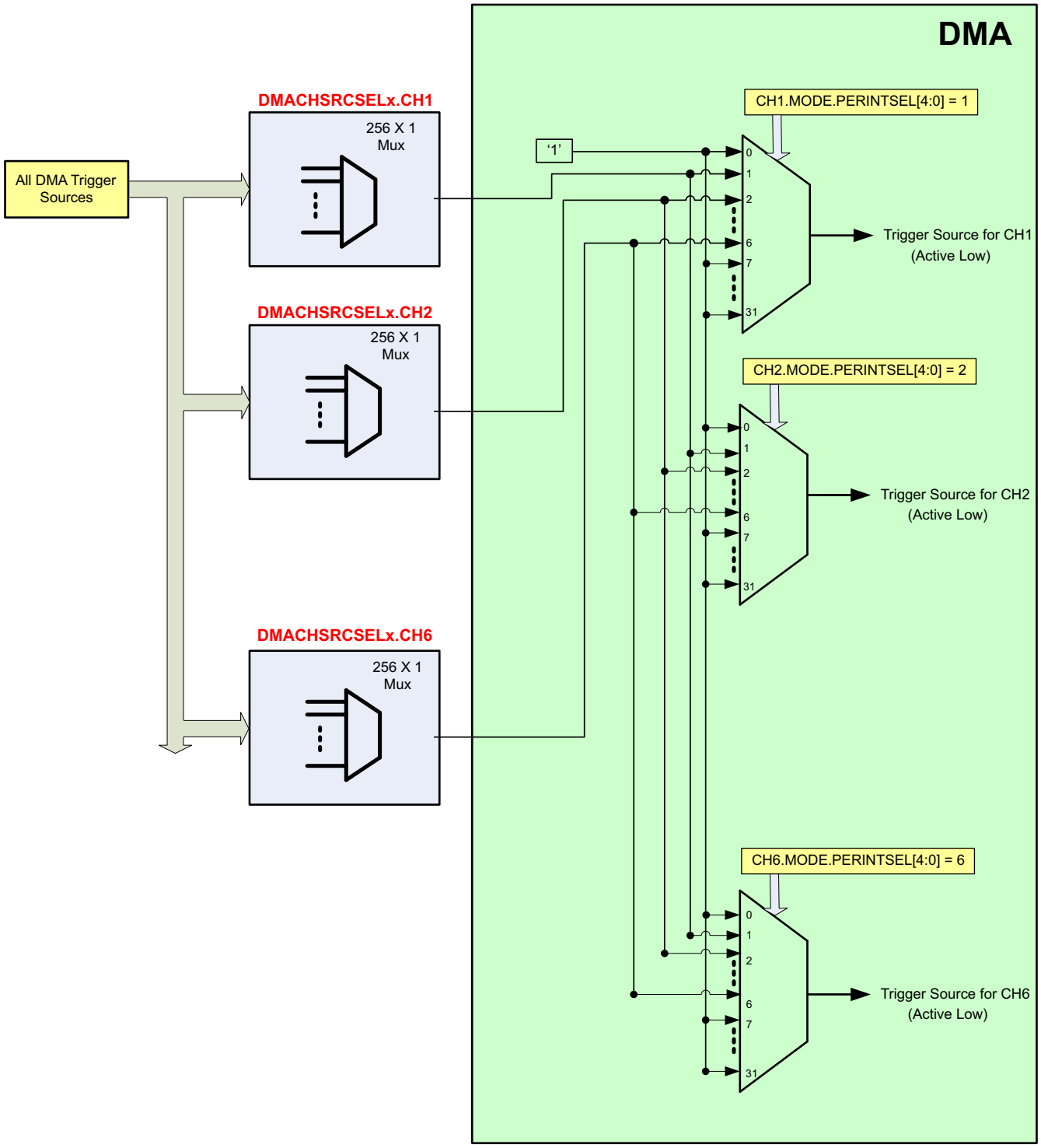
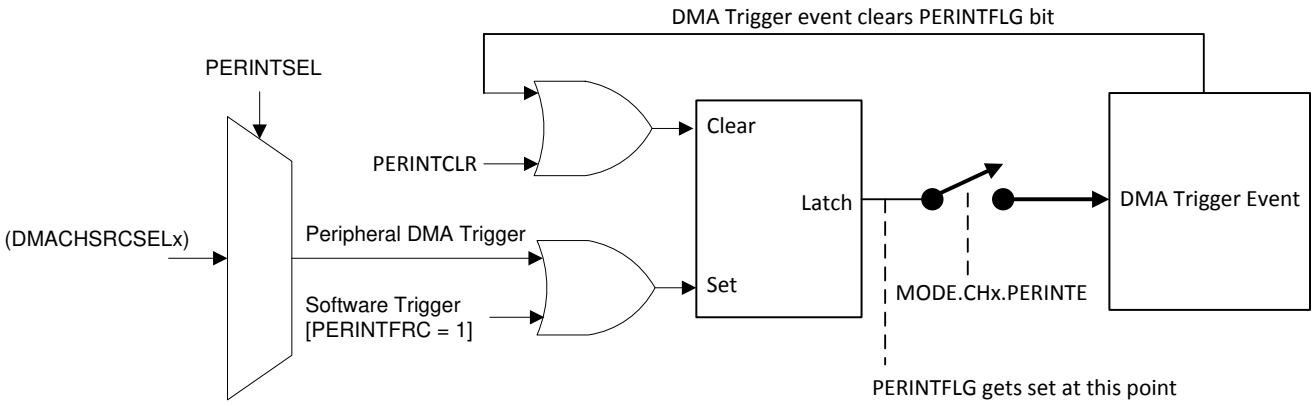



**Figure 5-3. DMA Trigger Architecture**



Copyright © 2024 Texas Instruments Incorporated




Note: See Figure 5-3.

**Figure 5-4. Peripheral Interrupt Trigger Input Diagram**

**Table 5-1. DMA Trigger Source Options**

| Select Index | Trigger Source |
|---|---|
| 0 | DMA_SOFTWARE_TRIGGER |
| 1 | ADCAINT1_DMA |
| 2 | ADCAINT2_DMA |
| 3 | ADCAINT3_DMA |
| 4 | ADCAINT4_DMA |
| 5 | ADCAEVT |
| 6 | ADCBINT1_DMA |
| 7 | ADCBINT2_DMA |
| 8 | ADCBINT3_DMA |
| 9 | ADCBINT4_DMA |
| 10 | ADCBEVT |
| 11 | ADCCINT1_DMA |
| 12 | ADCCINT2_DMA |
| 13 | ADCCINT3_DMA |
| 14 | ADCCINT4_DMA |
| 15 | ADCCEVT |
| 16 | ADCDINT1_DMA |
| 17 | ADCDINT2_DMA |
| 18 | ADCDINT3_DMA |
| 19 | ADCDINT4_DMA |
| 20 | ADCDEVT |
| 21-28 | Reserved |
| 29 | XINT1 |
| 30 | XINT2 |
| 31 | XINT3 |
| 32 | XINT4 |
| 33 | XINT5 |
| 34-35 | Reserved |
| 36 | EPWM1_SOCA |
| 37 | EPWM1_SOCB |
| 38 | EPWM2_SOCA |



Copyright © 2024 Texas Instruments Incorporated

**Table 5-1. DMA Trigger Source Options (continued)**

| Select Index | Trigger Source |
|---|---|
| 39 | EPWM2_SOCB |
| 40 | EPWM3_SOCA |
| 41 | EPWM3_SOCB |
| 42 | EPWM4_SOCA |
| 43 | EPWM4_SOCB |
| 44 | EPWM5_SOCA |
| 45 | EPWM5_SOCB |
| 46 | EPWM6_SOCA |
| 47 | EPWM6_SOCB |
| 48 | EPWM7_SOCA |
| 49 | EPWM7_SOCB |
| 50 | EPWM8_SOCA |
| 51 | EPWM8_SOCB |
| 52 | EPWM9_SOCA |
| 53 | EPWM9_SOCB |
| 54 | EPWM10_SOCA |
| 55 | EPWM10_SOCB |
| 56 | EPWM11_SOCA |
| 57 | EPWM11_SOCB |
| 58 | EPWM12_SOCA |
| 59 | EPWM12_SOCB |
| 60-67 | Reserved |
| 68 | CPU_TINT0 |
| 69 | CPU_TINT1 |
| 70 | CPU_TINT2 |
| 71 | MCBSPA_XEVT |
| 72 | MCBSPA_REVT |
| 73 | MCBSPB_XEVT |
| 74 | MCBSPB_REVT |
| 75-94 | Reserved |
| 95 | SD1FLT1_DRINT |
| 96 | SD1FLT2_DRINT |
| 97 | SD1FLT3_DRINT |
| 98 | SD1FLT4_DRINT |
| 99 | SD2FLT1_DRINT |
| 100 | SD2FLT2_DRINT |
| 101 | SD2FLT3_DRINT |
| 102 | SD2FLT4_DRINT |
| 103-108 | Reserved |
| 109 | SPIA_TXDMA |
| 110 | SPIA_RXDMA |
| 111 | SPIB_TXDMA |
| 112 | SPIB_RXDMA |
| 113 | SPIC_TXDMA |
| 114 | SPIC_RXDMA |



Copyright © 2024 Texas Instruments Incorporated

**Table 5-1. DMA Trigger Source Options (continued)**

| Select Index | Trigger Source |
|---|---|
| 115-126 | Reserved |
| 127 | CLB1_INT |
| 128 | CLB2_INT |
| 129 | CLB3_INT |
| 130 | CLB4_INT |
| 131-255 | Reserved |

### *5.2.3 DMA Bus*

The DMA bus architecture consists of a 32-bit address bus, a 32-bit data read bus, and a 32-bit data write bus. Memories and register locations connected to the DMA bus by way of interfaces that sometimes share resources with the CPU memory or peripheral bus. Arbitration rules are defined in Section 5.5.

## 5.3 Address Pointer and Transfer Control

The DMA state machine is, at the most basic level, two nested loops.

**Burst (Inner) Loop:**

The burst (inner) loop transfers a programmable number of words set by (BURST_SIZE + 1) register when a DMA channel trigger (Peripheral or Software trigger) is received. The BURST_SIZE register allows a maximum of 32 sixteen-bit words to be transferred in one burst. Each DMA channel supports both 16-bit or 32-bit word burst that can be controlled by MODE.DATASIZE bit field. Each DMA channel contains a shadowed address pointer for the source (SRC_ADDR_SHADOW) and the destination (DST_ADDR_SHADOW) address. At the beginning of each transfer, the shadowed version of each pointer is copied into the respective active (SRC_ADDR_ACTIVE or DST_ADDR_ACTIVE) register. During the burst loop, after each word is transferred, the signed value contained in the appropriate source or destination BURST_STEP register is added to the active register:

SRC_ADDR_ACTIVE = SRC_ADDR_ACTIVE + SRC_BURST_STEP

DST_ADDR_ACTIVE = DST_ADDR_ACTIVE + DST_BURST_STEP

The burst (inner) loop transfers a burst of data when a DMA Channel Trigger (Peripheral or Software trigger) is received.

**Transfer (Outer) Loop:**

The Transfer (outer) loop transfers a programmable number of bursts set by (TRANSFER_SIZE + 1) register for each channel. Since TRANSFER_SIZE is a 16-bit register, the total size of a transfer allowed is well beyond any practical requirement. During the transfer loop, after each burst is complete, there are two methods that can be used to modify the active address pointer:

**Method 1** (Default): When address wrapping is disabled (SRC_WRAP_SIZE or DST_WRAP_SIZE is greater than TRANSFER_SIZE), active address pointer is updated as shown below

SRC_ADDR_ACTIVE = SRC_ADDR_ACTIVE + SRC_TRANSFER_STEP

DST_ADDR_ACTIVE = DST_ADDR_ACTIVE + DST_TRANSFER_STEP



Copyright © 2024 Texas Instruments Incorporated

**Method 2:** Address wrapping gets enabled when SRC_WRAP_SIZE or DST_WRAP_SIZE is less than TRANSFER_SIZE. This allows the channel to wrap multiple times within a single transfer. When the number of bursts is equal to (SRC/DST_WRAP_SIZE + 1) register, the state machine modifies the active address pointers as:

SRC_BEG_ADDR_ACTIVE = SRC_BEG_ADDR_ACTIVE + SRC_WRAP_STEP

DST_BEG_ADDR_ACTIVE = DST_BEG_ADDR_ACTIVE + DST_WRAP_STEP

SRC_ADDR_ACTIVE = SRC_BEG_ADDR_ACTIVE

DST_ADDR_ACTIVE = DST_BEG_ADDR_ACTIVE

At the end of DMA transfer, DMA can have transferred (BURST_SIZE + 1) x (TRANSFER_SIZE + 1) words.

**OneShot Mode:**

OneShot mode is disabled by default.

When OneShot mode is disabled (MODE.CHx[ONESHOT] = 0), DMA transfers one burst [(BURST_SIZE + 1) words] of data each time a DMA Channel Trigger is received. After the burst is completed, the state machine moves on to the next pending channel in the priority scheme, even if another trigger for the channel just completed is pending. This feature keeps any single channel from monopolizing the DMA bus.

When OneShot mode is enabled (MODE.CHx[ONESHOT] = 1), DMA transfers all the bursts [(BURST_SIZE + 1) x (TRANSFER_SIZE + 1) words] on a single DMA channel trigger. Be careful when using this mode, since this can create a condition where one trigger uses up the majority of the DMA bandwidth.

**Continuous Mode:**

Continuous mode is disabled by default.

When Continuous mode is disabled (MODE.CHx[CONTINUOUS] = 0), DMA state machine disables channel after all bursts in a transfer loop (TRANSFER_COUNT = 0) are complete. The channel must be re-enabled by setting the RUN bit in the CONTROL register before another transfer can be started on that channel.

When Continuous mode is enabled (MODE.CHx[CONTINUOUS] = 1), DMA state machine keep channel active even after all bursts in a transfer loop (TRANSFER_COUNT = 0) are complete.

Each DMA channel can trigger an EPIE interrupt for each DMA transfer either at start of DMA transfer or end of DMA transfer using MODE.CHx[CHINTMODE] bit.

| | |
|---|---|
| **Source/Destination Address Pointers (SRC/DST_ADDR)** | The value written into the shadow register is the start address of the first location where data is read or written to.<br>At the beginning of a transfer the shadow register (SRC/DST_ADDR_SHADOW) is copied into the active register (SRC/DST_ADDR_ACTIVE). The active register performs as the current address pointer. |
| **Source/Destination Begin Address Pointers (SRC/DST_BEG_ADDR)** | This is the wrap pointer.<br>The value written into the shadow register (SRC/DST_BEG_ADDR_SHADOW) is loaded into the active register (SRC/DST_BEG_ADDR_ACTIVE) at the start of a transfer. On a wrap condition, the active register (SRC/DST_BEG_ADDR_ACTIVE) is incremented by the signed value in the appropriate SRC/DST_WRAP_STEP register prior to being loaded into the active register (SRC/DST_ADDR_ACTIVE). |



Copyright © 2024 Texas Instruments Incorporated

For each channel, the transfer process can be controlled with the following size values:

**Source and Destination Burst Size (BURST_SIZE)**

This specifies the number of words to be transferred in a burst.

This value is loaded into the BURST_COUNT register at the beginning of each burst. The BURST_COUNT decrements each word that is transferred and when the register reaches a zero value, the burst is complete, indicating that the next channel can be serviced. The behavior of the current channel is defined by the ONE_SHOT bit in the MODE register. The maximum size of the burst is dictated by the type of peripheral. For the ADC, the burst size can be all 16 registers (if all 16 registers are used). For RAM, the burst size can be up to the maximum allowed by the BURST_SIZE register, which is 32. See Table 5-2 to understand how BURST_SIZE register affects the number of 16-bit words transferred with respect to DATASIZE.

**Table 5-2. BURSTSIZE versus DATASIZE Behavior**

| BURSTSIZE | Number of 16-bit words transferred in | |
|---|---|---|
| | DataSize = 16-bit data | DataSize = 32-bit data |
| 0 | 1 | 2 |
| 1 | 2 | 2 |
| 2 | 3 | 4 |
| 3 | 4 | 4 |
| 4 | 5 | 6 |
| 5 | 6 | 6 |
| 6 | 7 | 8 |
| 7 | 8 | 8 |
| 8 | 9 | 10 |
| 9 | 10 | 10 |
| 10 | 11 | 12 |
| 11 | 12 | 12 |
| * | * | * |
| * | * | * |
| * | * | * |
| 30 | 31 | 32 |
| 31 | 32 | 32 |

**Source and Destination Transfer Size (TRANSFER_SIZE)**

This specifies the number of bursts to be transferred per CPU interrupt (if enabled).

Whether this interrupt is generated at the beginning or the end of the transfer is defined in the CHINTMODE bit in the MODE register. Whether the channel remains enabled or not after the transfer is completed is defined by the CONTINUOUS bit in the MODE register. The TRANSFER_SIZE register is loaded into the TRANSFER_COUNT register at the beginning of each transfer. The TRANSFER_COUNT register keeps track of how many bursts of data the channel has transferred and when the register reaches zero, the DMA transfer is complete.



Copyright © 2024 Texas Instruments Incorporated

**Source/Destination Wrap Size (SRC/ DST_WRAP_SIZE)**

This specifies the number of bursts to be transferred before the current address pointer wraps around to the beginning.

This feature is used to implement a circular addressing type function. This value is loaded into the appropriate SRC/DST_WRAP_COUNT register at the beginning of each transfer. The SRC/DST_WRAP_COUNT registers keep track of how many bursts of data the channel has transferred and when the registers reach zero, the wrap procedure is performed on the appropriate source or destination address pointer. A separate size and count register is allocated for source and destination pointers. To disable the wrap function, assign the value of these registers to be larger than the TRANSFER_SIZE.

**Note**

The value written to the SIZE registers is one less than the intended size. So, to transfer three 16-bit words, the value 2 can be placed in the SIZE register.

Regardless of the state of the DATASIZE bit, the value specified in the SIZE registers are for 16-bit addresses. So, to transfer three 32-bit words, the value 5 can be placed in the SIZE register.

For each source/destination pointer, the address changes can be controlled with the following step values:

**Source/Destination Burst Step (SRC/ DST_BURST_STEP)**

Within each burst transfer, the address source and destination step sizes are specified by these registers.

This value is a signed 2s compliment number so that the address pointer can be incremented or decremented as required. If no increment is desired, such as when accessing the data receive or transmit registers in a communication peripheral, the value of these registers can be set to zero.

**Source/Destination Transfer Step (SRC/ DST_TRANSFER_STEP)**

This specifies the address offset to start the next burst transfer after completing the current burst transfer.

This is used in cases where registers or data memory locations are spaced at constant intervals. This value is a signed 2s compliment number so that the address pointer can be incremented or decremented as required.

**Source/Destination Wrap Step (SRC/ DST_WRAP_STEP)**

When the wrap counter reaches zero, this value specifies the number of words to add/subtract from the SRC/DST_BEG_ADDR pointer and hence sets the new start address.

This implements a circular type of addressing mode, useful in many applications. This value is a signed 2s compliment number so that the address pointer can be incremented or decremented as required.

**Note**

Regardless of the state of the DATASIZE bit, the value specified in the STEP registers are for 16-bit addresses. So, to increment one 32-bit address, a value of 2 can be placed in these registers.

**Channel Interrupt Mode (CHINTMODE)**

This mode bit selects whether the DMA interrupt from the respective channel is generated at the beginning of a new transfer or at the end of the transfer.

If implementing a ping-pong buffer scheme with continuous mode of operation, then the interrupt can be generated at the beginning, just after the working registers are copied to the shadow set. If the DMA does not operate in continuous mode, then the interrupt is typically generated at the end when the transfer is complete.



Copyright © 2024 Texas Instruments Incorporated

All of the previous features and modes are shown in Figure 5-5. The following items are in reference to Figure 5-5.

- The *HALT* points represent where the channel halts operation when interrupted by a high priority channel 1 trigger, or when the HALT command is set, or when an emulation halt is issued and the FREE bit is cleared to 0.
- The SRC/DST_ADDR_ACTIVE registers are not affected by SRC/DST_BEG_ADDR_ACTIVE at the start of a transfer. SRC/DST_BEG_ADDR_ACTIVE only affects the SRC/DST_ADDR_ACTIVE registers on a wrap. Following is what happens when a transfer first starts:
  - SRC/DST_BEG_ADDR_SHADOW remains unchanged.
  - SRC/DST_ADDR_SHADOW remains unchanged.
  - SRC/DST_BEG_ADDR_ACTIVE = SRC/DST_BEG_ADDR_SHADOW
  - SRC/DST_ADDR_ACTIVE = SRC/DST_ADDR_SHADOW
- The active registers get updated when a wrap occurs. The shadow registers remain unchanged. Specifically:
  - SRC/DST_BEG_ADDR_SHADOW remains unchanged.
  - SRC/DST_ADDR_SHADOW remains unchanged.
  - SRC/DST_BEG_ADDR_ACTIVE += SRC/DST_WRAP_STEP
  - SRC/DST_ADDR_ACTIVE = SRC/DST_BEG_ADDR_ACTIVE
- The best way to remember this is:
  - The shadow registers never change except by software.
  - The active registers never change except by hardware, and a shadow register is only copied into the active register, never an active register by another name.



Copyright © 2024 Texas Instruments Incorporated

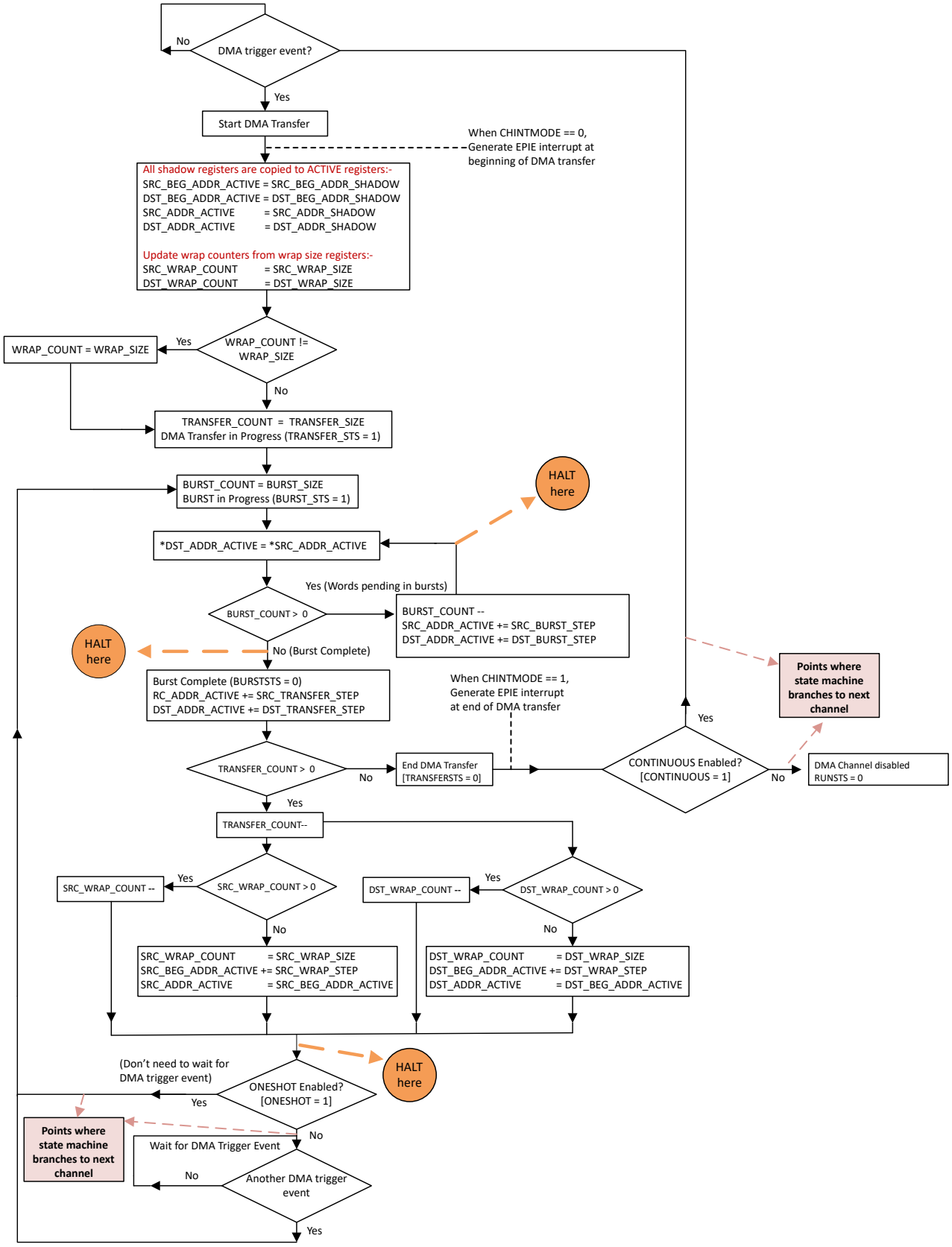



**Figure 5-5. DMA State Diagram**



Copyright © 2024 Texas Instruments Incorporated

## 5.4 Pipeline Timing and Throughput

In addition to the pipeline there are a few other behaviors of the DMA that affect the total throughput:

- A 1-cycle delay is added at the beginning of each burst
- A 1-cycle delay is added when returning from a CH1 high-priority interrupt
- Collisions with the CPU can add delay slots (see Section 5.5)
- 32-bit transfers run at double the speed of a 16-bit transfer (takes the same amount of time to transfer a 32-bit word as to transfer a 16-bit word)

For example, to transfer 128 16-bit words from GS0 RAM to GS3 RAM, a channel can be configured to transfer 8 bursts of 16 words/burst. This gives:

8 bursts * [(3 cycles/word * 16 words/burst) + 1] = 392 cycles

If instead the channel were configured to transfer the same amount of data 32 bits at a time (the word size is configured to 32 bits), the transfer can take:

8 bursts * [(3 cycles/word * 8 words/burst) + 1] = 200 cycles

The DMA module consists of a 3-stage pipeline as shown in Figure 5-6 and Figure 5-7.

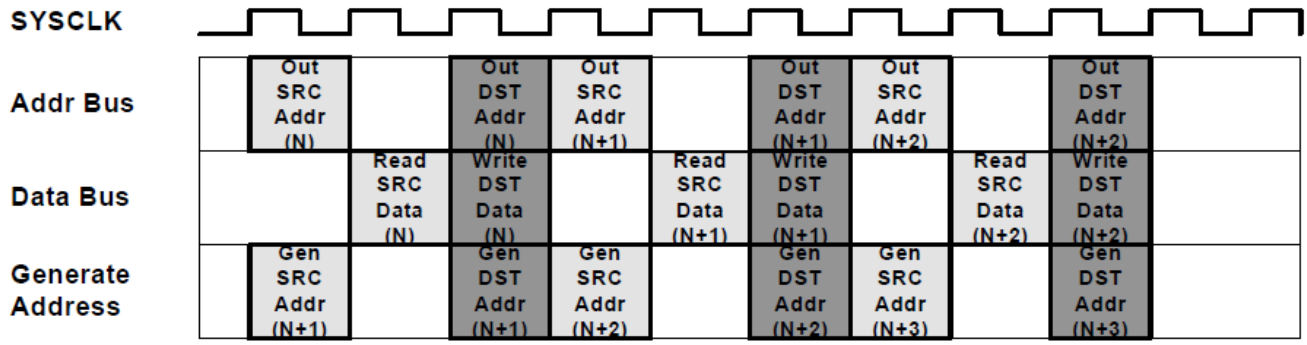



**Figure 5-6. 3-Stage Pipeline DMA Transfer**

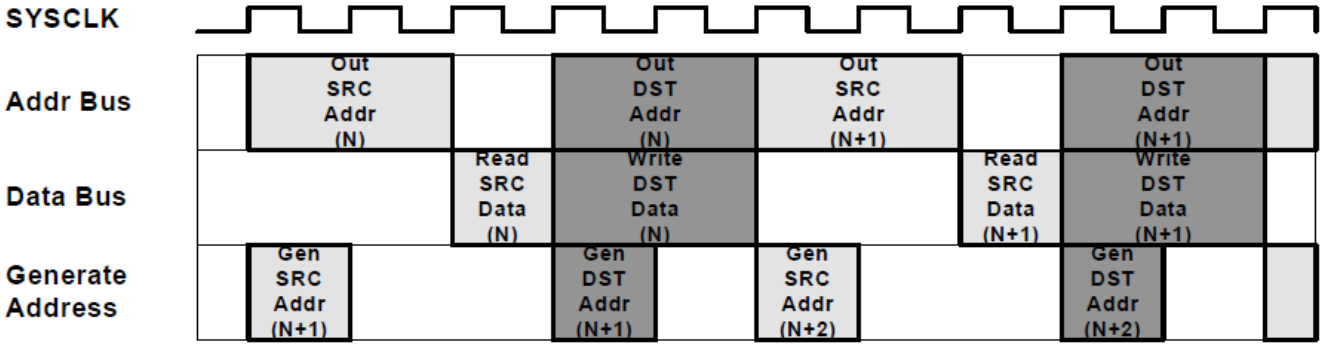



**Figure 5-7. 3-stage Pipeline with One Read Stall**



Copyright © 2024 Texas Instruments Incorporated

## 5.5 CPU and CLA Arbitration

Typically, DMA activity is independent of CPU and CLA activity. However, when the DMA and CPU (or CLA) try to access the same peripheral at the same time, an arbitration procedure is required to resolve the conflict. All instances of the same peripheral type conflict with each other. For instance, CAN-A and CAN-B conflict. Accesses to global shared RAM, across different instances, do not have this conflict. Different peripheral types can share a bus interface, which creates further opportunities for conflicts. These bus interfaces are:

- Peripheral frame 1: ePWM, eCAP, eQEP, CMPSS, DAC , SDFM
- Peripheral frame 2: PMBus, SPI

**Conflict Example:** The CLA is accessing DAC-A while the DMA is simultaneously accessing DAC-B.

**Conflict Example:** The CPU is accessing an SPI FIFO while the DMA is simultaneously accessing a PMBus register.

**Non-conflict Example**: The CPU is accessing a shared ePWM while the DMA is accessing an SPI.

**Non-conflict Example**: The CPU is accessing GS0 while the DMA is accessing GS1

The exception to all this is the ADC result registers, which are duplicated for each bus master. The CPU, DMA, and CLA can all simultaneously read these result registers with no stalls or arbitration needed for any master.

A DMA transfer consists of four phases: send source address, read source data, send destination address, and write destination data (see Section 5.4). Suppose CPU accesses a peripheral/memory causing conflict in middle of a DMA transfer, CPU is stalled till the current DMA access is complete and not until the completion of whole DMA transfer.

The following priority schemes are implemented for the various interfaces on the device:

- The fixed priority scheme for the peripheral frames is:
  - CLA/DMA Write
  - CLA/DMA Read
  - CPU Write
  - CPU Read
- The priority scheme for GSx RAM accesses is round-robin.
- The ADC results are duplicated for each CPU, CLA, and DMA so that no arbitration is needed when reading the result registers. This allows all masters to access the ADC result registers simultaneously without delay.

**Note**

If the CPU is performing a read-modify-write operation and the DMA performs a write to the same location, the DMA write can be lost if the operation occurs in between the CPU read and the CPU write. Avoid mixing CPU writes with DMA writes to the same locations.

Arbitration within DMA channels is based on a round-robin priority or Channel 1 high-priority scheme described in Section 5.6.



Copyright © 2024 Texas Instruments Incorporated

## 5.6 Channel Priority

Two priority schemes exist when determining channel priority: Round-robin mode and Channel 1 high-priority mode.

### *5.6.1 Round-Robin Mode*

In this mode, all channels have *equal* priority and each enabled channel is serviced in round-robin fashion as:

CH1 → CH2 → CH3 → CH4 → CH5 → CH6 → CH1 → CH2 → …

In the case above, after each channel has transferred a burst of words, the next channel is serviced. The user can specify the size of the burst for each channel. Once CH6 (or the last enabled channel) has been serviced, and no other channels are pending, the round-robin state machine enters an idle state.

From the idle state, channel 1 (if enabled) is always serviced first. However, if the DMA is currently processing another channel x, all other pending channels between x and the end of the round are serviced before CH1. All the channels are of *equal* priority. For instance, take an example where CH1, CH4, and CH5 are enabled in round-robin mode and CH4 is currently being processed. Then CH1 and CH5 both receive an interrupt trigger from the respective peripherals before CH4 completes. CH1 and CH5 are now both pending. When CH4 completes the burst, CH5 is serviced next. Only after CH5 completes is CH1 serviced. Upon completion of CH1, if there are no more channels pending, the round-robin state machine enters an idle state.

A more complicated example is:

- Assume all channels are enabled, and the DMA is in an idle state,
- Initially a trigger occurs on CH1, CH3, and CH5 on the same cycle,
- When the CH1 burst transfer starts, requests from CH3 and CH5 are pending,
- Before completion of the CH1 burst, the DMA receives a request from CH2. Now the pending requests are from CH2, CH3, and CH5,
- After completing the CH1 burst, CH2 is serviced since this channel is next in the round-robin scheme after CH1.
- After the burst from CH2 is finished, the CH3 burst is serviced, followed by CH5 burst.
- Now while the CH5 burst is being serviced, the DMA receives a request from CH1, CH3, and CH6.
- The burst from CH6 starts after the completion of the CH5 burst, since this channel is the next channel after CH5 in the round-robin scheme.
- This is followed by the CH1 burst and then the CH3 burst
- After the CH3 burst finishes, assuming no more triggers have occurred, the round-robin state machine enters an idle state.

The round-robin state machine can be reset to the idle state using the DMACTRL[PRIORITYRESET] bit.



Copyright © 2024 Texas Instruments Incorporated

### 5.6.2 Channel 1 High-Priority Mode

In this mode, Channel 1 has high priority over all the other channels. Channels 2 to 6 have equal priority and each enabled channel is serviced in a round-robin fashion.

| | |
|---|---|
| Higher priority: | CH1 |
| Lower priority: | CH2 → CH3 → CH4 → CH5 → CH6 → CH2 → … |

Given an example where CH1, CH4, and CH5 are enabled in Channel 1 high-priority mode and CH4 is currently being processed. Then CH1 and CH5 both receive an interrupt trigger from the respective peripherals before CH4 completes. CH1 and CH5 are now both pending. When the current CH4 word transfer is completed, regardless of whether the DMA has completed the entire CH4 burst, CH4 execution is suspended and CH1 is serviced. After the CH1 burst completes, CH4 resumes execution.

Upon completion of CH4, CH5 is serviced. After CH5 completes, if there are no more channels pending, the round-robin state machine enters an idle state.

Typically Channel 1 is used in this mode for the ADC, since the data rate is so high. However, Channel 1 high-priority mode can be used in conjunction with any peripheral.

**Note**

High-priority mode and ONESHOT mode cannot be used at the same time on Channel 1. Other channels can use ONESHOT mode when Channel 1 is in high-priority mode.

## 5.7 Overrun Detection Feature

The DMA contains overrun detection logic. When a peripheral event trigger is received by the DMA, the PERINTFLG bit in the CONTROL register is set, pending the channel to the DMA state machine. When the burst for that channel is started, the PERINTFLG is cleared. If however, between the time that the PERINTFLG bit is set by an event trigger and cleared by the start of the burst, an additional event trigger arrives, the second trigger is lost. This condition sets the OVRFLG bit in the CONTROL register as in Figure 5-8. If the overrun interrupt is enabled, the channel interrupt is generated to the PIE module.

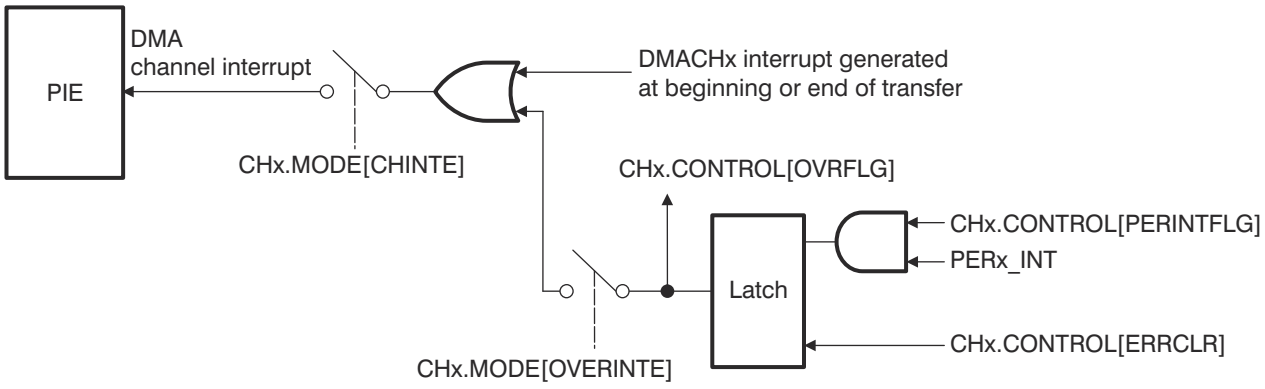



**Figure 5-8. Overrun Detection Logic**



Copyright © 2024 Texas Instruments Incorporated

## 5.8 Software

### *5.8.1 DMA Examples*

NOTE: These examples are located in the C2000Ware installation at the following location: C2000Ware_VERSION#/driverlib/DEVICE_GPN/examples/CORE_IF_MULTICORE/dma

Cloud access to these examples is available at the following link: dev.ti.com C2000Ware Examples.

#### 5.8.1.1 DMA GSRAM Transfer (dma_ex1_gsram_transfer)

FILE: dma_ex1_gsram_transfer.c

This example uses one DMA channel to transfer data from a buffer in RAMGS0 to a buffer in RAMGS1. The example sets the DMA channel PERINTFRC bit repeatedly until the transfer of 16 bursts (where each burst is 8 16-bit words) has been completed. When the whole transfer is complete, it will trigger the DMA interrupt.

: This example project has support for migration across our C2000 device families. If you are wanting to build this project from launchpad or controlCARD, please specify in the .syscfg file the board you're using. At any time you can select another device to migrate this example. *Watch Variables*

- *sData* - Data to send
- *rData* - Received data

#### 5.8.1.2 DMA Transfer Shared Peripheral - C28X_DUAL

FILE: dma_ex1_shared_periph_cpu1.c

This example shows how to initiate a DMA transfer on CPU1 from a shared peripheral which is owned by CPU2. In this specific example, a timer ISR is used on CPU2 to initiate a SPI transfer which will trigger the CPU1 DMA. CPU1's DMA will then in turn update the ePWM1 CMPA value for the PWM which it owns. The PWM output can be observed on the GPIO pins. It is recommended to run the c28x1 core first, followed by the C28x2 core.

*Watch Pins*

- GPIO0 and GPIO1 - ePWM output can be viewed with oscilloscope

#### 5.8.1.3 DMA Transfer for Shared Peripheral Example (CPU2) - C28X_DUAL

FILE: dma_ex1_shared_periph_cpu2.c

This example shows how to initiate a DMA transfer on CPU1 from a shared peripheral that is owned by CPU2. In this specific example, a timer ISR is used on CPU2 to initiate a SPI transfer that triggers the CPU1 DMA. CPU1 DMA then updates the ePWM1 CMPA value for the PWM that the DMA owns. The PWM output can be observed on the GPIO pins. It is recommended to run the c28x1 core first, followed by the C28x2 core.

#### 5.8.1.4 DMA GSRAM Transfer (dma_ex2_gsram_transfer)

FILE: dma_ex2_gsram_transfer.c

This example uses one DMA channel to transfer data from a buffer in RAMGS0 to a buffer in RAMGS1. The example sets the DMA channel PERINTFRC bit repeatedly until the transfer of 16 bursts (where each burst is 8 16-bit words) has been completed. When the whole transfer is complete, it will trigger the DMA interrupt.

*Watch Variables*

- *sData* - Data to send
- *rData* - Received data



Copyright © 2024 Texas Instruments Incorporated

## 5.9 DMA Registers

This section describes the Direct Memory Access Registers.

### *5.9.1 DMA Base Addresses*

**Table 5-3. DMA Base Address Table**

| Device Registers | Register Name | Start Address | End Address |
|---|---|---|---|
| DmaRegs | DMA_REGS | 0x0000_1000 | 0x0000_11FF |



Copyright © 2024 Texas Instruments Incorporated

### 5.9.2 DMA_REGS Registers

Table 5-4 lists the memory-mapped registers for the DMA_REGS registers. All register offset addresses not listed in Table 5-4 should be considered as reserved locations and the register contents should not be modified.

**Table 5-4. DMA_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | DMACTRL | DMA Control Register | EALLOW | Go |
| 1h | DEBUGCTRL | Debug Control Register | EALLOW | Go |
| 4h | PRIORITYCTRL1 | Priority Control 1 Register | EALLOW | Go |
| 6h | PRIORITYSTAT | Priority Status Register | EALLOW | Go |

Complex bit access types are encoded to fit into small table cells. Table 5-5 shows the codes that are used for access types in this section.

**Table 5-5. DMA_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| R-0 | R<br>-0 | Read<br>Returns 0s |
| Write Type | | |
| W | W | Write |
| W1S | W<br>1S | Write<br>1 to set |
| Reset or Default Value | | |
| *-n* | | Value after reset or the default value |



Copyright © 2024 Texas Instruments Incorporated

### 5.9.2.1 DMACTRL Register (Offset = 0h) [Reset = 0000h]

DMACTRL is shown in Figure 5-9 and described in Table 5-6.

Return to the Summary Table.

DMA Control Register

**Figure 5-9. DMACTRL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | PRIORITYRESET | HARDRESET |
| R-0h | | | | | | R-0/W1S-0h | R-0/W1S-0h |

**Table 5-6. DMACTRL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-2 | RESERVED | R | 0h | Reserved |
| 1 | PRIORITYRESET | R-0/W1S | 0h | The priority reset bit resets the round-robin state machine when a 1 is written. Service starts from the first enabled channel. Writes of 0 are ignored and this bit always reads back a 0.<br>When a 1 is written to this bit, any pending burst transfer completes before resetting the channel priority machine. If CH1 is configured as a high-priority channel, and this bit is<br>written to while CH1 is servicing a burst, both the CH1 burst and the next pending low-priority burst are completed before the state machine is reset.<br>If CH1 is high-priority, the state machine restarts from CH2 (or the next highest enabled channel).<br>Reset type: SYSRSn |
| 0 | HARDRESET | R-0/W1S | 0h | Writing a 1 to the hard reset bit resets the whole DMA and aborts any current access (similar to applying a device reset). Writes of 0 are ignored and this bit always reads back a 0.<br>For a soft reset, a bit is provided for each channel to perform a gentler reset. Refer to the channel control registers.<br>When writing to this bit, there is a one cycle delay before it takes effect. Hence, a one-cycle delay (such as a NOP instruction) is required in software before attempting to access any other DMA register.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

#### 5.9.2.2 DEBUGCTRL Register (Offset = 1h) [Reset = 0000h]

DEBUGCTRL is shown in Figure 5-10 and described in Table 5-7.

Return to the Summary Table.

Debug Control Register

**Figure 5-10. DEBUGCTRL Register**

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

**Table 5-7. DEBUGCTRL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | FREE | R/W | 0h | Emulation Control<br>This bit specifies the action when an emulation halt event occurs.<br>Reset type: SYSRSn<br>0h (R/W) = The DMA completes the current read-write operation, then halts.<br>1h (R/W) = The DMA continues running during an emulation halt. |
| 14-0 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 5.9.2.3 PRIORITYCTRL1 Register (Offset = 4h) [Reset = 0000h]

PRIORITYCTRL1 is shown in Figure 5-11 and described in Table 5-8.

Return to the Summary Table.

Priority Control 1 Register

**Figure 5-11. PRIORITYCTRL1 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | | CH1PRIORITY |
| R-0h | | | | | | | R/W-0h |

**Table 5-8. PRIORITYCTRL1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-1 | RESERVED | R | 0h | Reserved |
| 0 | CH1PRIORITY | R/W | 0h | DMA Channel 1 Priority<br>This bit selects whether CH1 has high priority or not. The priority can only be changed when all channels are disabled. A priority reset should be performed before restarting channels after changing priority<br>Reset type: SYSRSn<br>0h (R/W) = CH1 has the same priority as the other channels<br>1h (R/W) = CH1 has a higher priority than the other channels |



Copyright © 2024 Texas Instruments Incorporated

### 5.9.2.4 PRIORITYSTAT Register (Offset = 6h) [Reset = 0000h]

PRIORITYSTAT is shown in Figure 5-12 and described in Table 5-9.

Return to the Summary Table.

Priority Status Register

**Figure 5-12. PRIORITYSTAT Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | ACTIVESTS_SHADOW | | | RESERVED | ACTIVESTS | | |
| R-0h | R-0h | | | R-0h | R-0h | | |

**Table 5-9. PRIORITYSTAT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-7 | RESERVED | R | 0h | Reserved |
| 6-4 | ACTIVESTS_SHADOW | R | 0h | Active Channel Status Shadow<br>These bits are only meaningful when CH1 is in high-priority mode. When CH1 is serviced, the ACTIVESTS bits are copied to the shadow bits and indicate which channel was interrupted by CH1. When CH1 service is completed, the shadow bits are copied back to the ACTIVESTS bits. If this bit field is zero or the same as the ACTIVESTS bit field, then no channel is pending due to a CH1 interrupt. When CH1 is not a higher priority channel, these bits should be ignored.<br>Reset type: SYSRSn<br>0h (R/W) = No channel is active<br>1h (R/W) = CH 1<br>2h (R/W) = CH 2<br>3h (R/W) = CH 3<br>4h (R/W) = CH 4<br>5h (R/W) = CH 5<br>6h (R/W) = CH 6<br>7h (R/W) = Reserved |
| 3 | RESERVED | R | 0h | Reserved |
| 2-0 | ACTIVESTS | R | 0h | Active Channel Status<br>These bits indicate which channel (if any) is currently active or performing a transfer.<br>Reset type: SYSRSn<br>0h (R/W) = No channel is active<br>1h (R/W) = CH 1<br>2h (R/W) = CH 2<br>3h (R/W) = CH 3<br>4h (R/W) = CH 4<br>5h (R/W) = CH 5<br>6h (R/W) = CH 6<br>7h (R/W) = Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 5.9.3 DMA_CH_REGS Registers

Table 5-10 lists the memory-mapped registers for the DMA_CH_REGS registers. All register offset addresses not listed in Table 5-10 should be considered as reserved locations and the register contents should not be modified.

**Table 5-10. DMA_CH_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | MODE | Mode Register | EALLOW | Go |
| 1h | CONTROL | Control Register | EALLOW | Go |
| 2h | BURST_SIZE | Burst Size Register | EALLOW | Go |
| 3h | BURST_COUNT | Burst Count Register | EALLOW | Go |
| 4h | SRC_BURST_STEP | Source Burst Step Register | EALLOW | Go |
| 5h | DST_BURST_STEP | Destination Burst Step Register | EALLOW | Go |
| 6h | TRANSFER_SIZE | Transfer Size Register | EALLOW | Go |
| 7h | TRANSFER_COUNT | Transfer Count Register | EALLOW | Go |
| 8h | SRC_TRANSFER_STEP | Source Transfer Step Register | EALLOW | Go |
| 9h | DST_TRANSFER_STEP | Destination Transfer Step Register | EALLOW | Go |
| Ah | SRC_WRAP_SIZE | Source Wrap Size Register | EALLOW | Go |
| Bh | SRC_WRAP_COUNT | Source Wrap Count Register | EALLOW | Go |
| Ch | SRC_WRAP_STEP | Source Wrap Step Register | EALLOW | Go |
| Dh | DST_WRAP_SIZE | Destination Wrap Size Register | EALLOW | Go |
| Eh | DST_WRAP_COUNT | Destination Wrap Count Register | EALLOW | Go |
| Fh | DST_WRAP_STEP | Destination Wrap Step Register | EALLOW | Go |
| 10h | SRC_BEG_ADDR_SHADOW | Source Begin Address Shadow Register | EALLOW | Go |
| 12h | SRC_ADDR_SHADOW | Source Address Shadow Register | EALLOW | Go |
| 14h | SRC_BEG_ADDR_ACTIVE | Source Begin Address Active Register | EALLOW | Go |
| 16h | SRC_ADDR_ACTIVE | Source Address Active Register | EALLOW | Go |
| 18h | DST_BEG_ADDR_SHADOW | Destination Begin Address Shadow Register | EALLOW | Go |
| 1Ah | DST_ADDR_SHADOW | Destination Address Shadow Register | EALLOW | Go |
| 1Ch | DST_BEG_ADDR_ACTIVE | Destination Begin Address Active Register | EALLOW | Go |
| 1Eh | DST_ADDR_ACTIVE | Destination Address Active Register | EALLOW | Go |

Complex bit access types are encoded to fit into small table cells. Table 5-11 shows the codes that are used for access types in this section.

**Table 5-11. DMA_CH_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| R-0 | R<br>-0 | Read<br>Returns 0s |
| Write Type | | |
| W | W | Write |
| W1S | W<br>1S | Write<br>1 to set |
| Reset or Default Value | | |
| *-n* | | Value after reset or the default value |



Copyright © 2024 Texas Instruments Incorporated

#### 5.9.3.1 MODE Register (Offset = 0h) [Reset = 0000h]

MODE is shown in Figure 5-13 and described in Table 5-12.

Return to the Summary Table.

Mode Register

**Figure 5-13. MODE Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| CHINTE | DATASIZE | RESERVED | RESERVED | CONTINUOUS | ONESHOT | CHINTMODE | PERINTE |
| 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 |
| OVRINTE | RESERVED | | PERINTSEL | | | | |
| R/W-0h | R-0h | | R/W-0h | | | | |

**Table 5-12. MODE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | CHINTE | R/W | 0h | Channel Interrupt Enable Bit<br>This bit enables the DMA channel's CPU interrupt.<br>Reset type: SYSRSn<br>0h (R/W) = Interrupt disabled<br>1h (R/W) = Interrupt enabled |
| 14 | DATASIZE | R/W | 0h | Data Size Mode Bit<br>This bit determines whether the DMA channel transfers 16 bits or 32 bits of data per read/write operation. Regardless of this setting, all data lengths and offsets in other DMA registers refer to 16- bit words. The pointer step increments must be configured to accomodate 32-bit words.<br>Reset type: SYSRSn<br>0h (R/W) = 16-bit data transfer size<br>1h (R/W) = 32-bit data transfer size |
| 13 | RESERVED | R/W | 0h | Reserved |
| 12 | RESERVED | R/W | 0h | Reserved |
| 11 | CONTINUOUS | R/W | 0h | Continuous Mode Bit<br>If this bit is set to 1, then the channel re-initializes when TRANSFER_COUNT is zero and waits for the next event trigger. Otherwise, the DMA stops and clears the RUNSTS bit.<br>Reset type: SYSRSn |
| 10 | ONESHOT | R/W | 0h | One Shot Mode<br>If this bit is set to 1, each peripheral event trigger causes the channel to perform an entire transfer. Otherwise, the channel only performs one burst per trigger.<br>Reset type: SYSRSn |
| 9 | CHINTMODE | R/W | 0h | Channel Interrupt Generation Mode<br>This bit specifies when the DMA channel generates a CPU interrupt for a transfer.<br>Reset type: SYSRSn<br>0h (R/W) = Generate interrupt at beginning of new transfer<br>1h (R/W) = Generate interrupt at end of transfer. |
| 8 | PERINTE | R/W | 0h | Peripheral Event Trigger Enable<br>This bit enables peripheral event triggers on the DMA channel.<br>Reset type: SYSRSn<br>0h (R/W) = Peripheral event trigger disabled. Neither the selected peripheral nor software can start a DMA burst.<br>1h (R/W) = Peripheral event trigger enabled. |



Copyright © 2024 Texas Instruments Incorporated

**Table 5-12. MODE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 7 | OVRINTE | R/W | 0h | Overflow Interrupt Enable<br>The bit determines whether the DMA module generates a CPU interrupt when it detects an overflow event.<br>Reset type: SYSRSn<br>0h (R/W) = Overflow interrupt disabled<br>1h (R/W) = Overflow interrupt enabled |
| 6-5 | RESERVED | R | 0h | Reserved |
| 4-0 | PERINTSEL | R/W | 0h | Peripheral Event Trigger Source Select<br>These are legacy bits and should be set to the channel number. The actual source selection is done via the DMACHSRCSELn registers, which are part of the DMA_CLA_SRC_SEL_REGS group.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 5.9.3.2 CONTROL Register (Offset = 1h) [Reset = 0000h]

CONTROL is shown in Figure 5-14 and described in Table 5-13.

Return to the Summary Table.

Control Register

**Figure 5-14. CONTROL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | OVRFLG | RUNSTS | BURSTSTS | TRANSFERSTS | RESERVED | RESERVED | PERINTFLG |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ERRCLR | RESERVED | RESERVED | PERINTCLR | PERINTFRC | SOFTRESET | HALT | RUN |
| R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h |

**Table 5-13. CONTROL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RESERVED | R | 0h | Reserved |
| 14 | OVRFLG | R | 0h | Overflow Flag<br>This bit indicates that a peripheral event trigger was received while PERINTFLG was already set. It can be cleared by writing to the ERRCLR bit.<br>Reset type: SYSRSn<br>0h (R/W) = No overflow detected<br>1h (R/W) = Overflow detected |
| 13 | RUNSTS | R | 0h | Run Status Flag<br>This bit indicates that the DMA channel is ready to respond to peripheral event triggers. This bit is set when a 1 is written to the RUN bit. It is cleared when a transfer completes (TRANSFER_COUNT = 0) and continuous mode is disabled, or when the HARDRESET, SOFTRESET, or HALT bit is set.<br>Reset type: SYSRSn<br>0h (R/W) = The channel is disabled<br>1h (R/W) = The channel is enabled |
| 12 | BURSTSTS | R | 0h | Burst Status Flag<br>This bit is set when a DMA burst begins. The BURST_COUNT is set to the BURST_SIZE. This bit is cleared when BURST_COUNT reaches zero, or when the HARDRESET or SOFTRESET bit is set.<br>Reset type: SYSRSn<br>0h (R/W) = No burst activity<br>1h (R/W) = The DMA is currently servicing or suspending a burst transfer from this channel |
| 11 | TRANSFERSTS | R | 0h | Transfer Status Flag<br>This bit is set when a DMA transfer begins. The address registers are copied to the shadow set and the TRANSFER_COUNT is set to the TRANSFER_SIZE. This bit is cleared when TRANSFER_COUNT reaches zero, or when the HARDRESET or SOFTRESET bit is set.<br>Reset type: SYSRSn<br>0h (R/W) = No transfer activity<br>1h (R/W) = The channel is currently in the middle of a transfer regardless of whether a burst of data is actively being transferred or not |
| 10 | RESERVED | R | 0h | Reserved |
| 9 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 5-13. CONTROL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 8 | PERINTFLG | R | 0h | Peripheral Event Trigger Flag<br>This bit indicates whether a peripheral event trigger has arrived. This bit is automatically cleared when the first burst transfer begins.<br>Reset type: SYSRSn<br>0h (R/W) = Waiting for event trigger<br>1h (R/W) = Event trigger pending |
| 7 | ERRCLR | R-0/W1S | 0h | Clear Error<br>Writing a 1 to this bit will clear the OVRFLG bit. This is normally done when initializing the DMA module or if an overflow condition is detected. If an overflow event occurs at the same time this bit is set, the overrun has priority and the OVRFLG bit is set.<br>Reset type: SYSRSn |
| 6 | RESERVED | R-0/W1S | 0h | Reserved |
| 5 | RESERVED | R-0/W1S | 0h | Reserved |
| 4 | PERINTCLR | R-0/W1S | 0h | Clear Peripheral Event Trigger<br>Writing a 1 to this bit clears PERINTFLG, which cancels a pending event trigger. This is normally done when initializing the DMA module. If an event trigger arrives at the same time this bit is set, the trigger has priority and PERINTFLG is set.<br>Reset type: SYSRSn |
| 3 | PERINTFRC | R-0/W1S | 0h | Force Peripheral Event Trigger<br>If the PERINTE bit of the MODE register is set, writing a 1 to this bit sets PERINTFLG, which triggers a DMA burst. This bit can be used to start a DMA transfer in software.<br>Reset type: SYSRSn |
| 2 | SOFTRESET | R-0/W1S | 0h | Channel Soft Reset<br>Writing a 1 to this bit places the channel into its default state after the current read/write access has completed:<br>RUNSTS = 0<br>TRANSFERSTS = 0<br>BURSTSTS = 0<br>BURST_COUNT = 0<br>TRANSFER_COUNT = 0<br>SRC_WRAP_COUNT = 0<br>DST_WRAP_COUNT = 0<br>When writing to this bit, there is a one cycle delay before it takes effect. Hence, a one-cycle delay (such as a NOP instruction) is required in software before attempting to access any other DMA register.<br>Reset type: SYSRSn |
| 1 | HALT | R-0/W1S | 0h | Halt Channel<br>Writing a 1 to this bit halts the DMA channel in its current state after any ongoing read/write access has completed.<br>Reset type: SYSRSn |
| 0 | RUN | R-0/W1S | 0h | Run Channel<br>Writing a 1 to this bit enables the DMA channel and sets the RUNSTS bit to 1. This bit is also used to resume after a channel halt.<br>The RUN bit is typically used to start the DMA channel after configuration. The channel will then wait for the first peripheral event trigger (PERINTFLG == 1) to start a burst.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

#### 5.9.3.3 BURST_SIZE Register (Offset = 2h) [Reset = 0000h]

BURST_SIZE is shown in Figure 5-15 and described in Table 5-14.

Return to the Summary Table.

Burst Size Register

**Figure 5-15. BURST_SIZE Register**

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

**Table 5-14. BURST_SIZE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-5 | RESERVED | R | 0h | Reserved |
| 4-0 | BURSTSIZE | R/W | 0h | These bits specify the burst size in 16-bit words. The actual size is equal to BURSTSIZE + 1.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 5.9.3.4 BURST_COUNT Register (Offset = 3h) [Reset = 0000h]

BURST_COUNT is shown in Figure 5-16 and described in Table 5-15.

Return to the Summary Table.

Burst Count Register

**Figure 5-16. BURST_COUNT Register**

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

**Table 5-15. BURST_COUNT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-5 | RESERVED | R | 0h | Reserved |
| 4-0 | BURSTCOUNT | R | 0h | These bits indicate the number of words left in the current burst.<br>Reset type: SYSRSn<br>0h (R/W) = 0 word left in a burst<br>1h (R/W) = 1 word left in a burst<br>2h (R/W) = 2 word left in a burst<br>3h (R/W) = 3 word left in a burst<br>4h (R/W) = 4 word left in a burst<br>5h (R/W) = 5 word left in a burst<br>6h (R/W) = 6 word left in a burst<br>7h (R/W) = 7 word left in a burst<br>8h (R/W) = 8 word left in a burst<br>9h (R/W) = 9 word left in a burst<br>Ah (R/W) = 10 word left in a burst<br>Bh (R/W) = 11 word left in a burst<br>Ch (R/W) = 12 word left in a burst<br>Dh (R/W) = 13 word left in a burst<br>Eh (R/W) = 14 word left in a burst<br>Fh (R/W) = 15 word left in a burst<br>10h (R/W) = 16 word left in a burst<br>11h (R/W) = 17 word left in a burst<br>12h (R/W) = 18 word left in a burst<br>13h (R/W) = 19 word left in a burst<br>14h (R/W) = 20 word left in a burst<br>15h (R/W) = 21 word left in a burst<br>16h (R/W) = 22 word left in a burst<br>17h (R/W) = 23 word left in a burst<br>18h (R/W) = 24 word left in a burst<br>19h (R/W) = 25 word left in a burst<br>1Ah (R/W) = 26 word left in a burst<br>1Bh (R/W) = 27 word left in a burst<br>1Ch (R/W) = 28 word left in a burst<br>1Dh (R/W) = 29 word left in a burst<br>1Eh (R/W) = 30 word left in a burst<br>1Fh (R/W) = 31 word left in a burst |



Copyright © 2024 Texas Instruments Incorporated

#### 5.9.3.5 SRC_BURST_STEP Register (Offset = 4h) [Reset = 0000h]

SRC_BURST_STEP is shown in Figure 5-17 and described in Table 5-16.

Return to the Summary Table.

Source Burst Step Register

**Figure 5-17. SRC_BURST_STEP Register**

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

**Table 5-16. SRC_BURST_STEP Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | SRCBURSTSTEP | R/W | 0h | These bits specify the change in the source address after each word in a burst. The size must be a 16-bit two's complement value between -4096 and 4095 (inclusive). This value is added to the source address after each read/write operation in a burst.<br>Reset type: SYSRSn<br>0h (R/W) = No address change<br>1h (R/W) = Add 1 to the address<br>2h (R/W) = Add 2 to the address<br>FFEh (R/W) = Add 4094 to the address<br>FFFh (R/W) = Add 4095 to the address<br>F000h (R/W) = Subtract 4096 from the address<br>F001h (R/W) = Subtract 4095 from the address<br>FFFEh (R/W) = Subtract 2 from the address<br>FFFFh (R/W) = Subtract 1 from the address |



Copyright © 2024 Texas Instruments Incorporated

### 5.9.3.6 DST_BURST_STEP Register (Offset = 5h) [Reset = 0000h]

DST_BURST_STEP is shown in Figure 5-18 and described in Table 5-17.

Return to the Summary Table.

Destination Burst Step Register

**Figure 5-18. DST_BURST_STEP Register**

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

**Table 5-17. DST_BURST_STEP Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | DSTBURSTSTEP | R/W | 0h | These bits specify the change in the destination address after each word in a burst. The size must be a 16-bit two's complement value between -4096 and 4095 (inclusive). This value is added to the destination address after each read/write operation in a burst.<br>Reset type: SYSRSn<br>0h (R/W) = No address change<br>1h (R/W) = Add 1 to the address<br>2h (R/W) = Add 2 to the address<br>FFEh (R/W) = Add 4094 to the address<br>FFFh (R/W) = Add 4095 to the address<br>F000h (R/W) = Subtract 4096 from the address<br>F001h (R/W) = Subtract 4095 from the address<br>FFFEh (R/W) = Subtract 2 from the address<br>FFFFh (R/W) = Subtract 1 from the address |



Copyright © 2024 Texas Instruments Incorporated

### 5.9.3.7 TRANSFER_SIZE Register (Offset = 6h) [Reset = 0000h]

TRANSFER_SIZE is shown in Figure 5-19 and described in Table 5-18.

Return to the Summary Table.

Transfer Size Register

**Figure 5-19. TRANSFER_SIZE Register**

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

**Table 5-18. TRANSFER_SIZE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | TRANSFERSIZE | R/W | 0h | These bits specify the transfer size in bursts. The actual size is equal to TRANSFERSIZE + 1.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

#### 5.9.3.8 TRANSFER_COUNT Register (Offset = 7h) [Reset = 0000h]

TRANSFER_COUNT is shown in Figure 5-20 and described in Table 5-19.

Return to the Summary Table.

Transfer Count Register

**Figure 5-20. TRANSFER_COUNT Register**

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

**Table 5-19. TRANSFER_COUNT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | TRANSFERCOUNT | R | 0h | These bits indicate the number of bursts left in the current transfer.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

#### 5.9.3.9 SRC_TRANSFER_STEP Register (Offset = 8h) [Reset = 0000h]

SRC_TRANSFER_STEP is shown in Figure 5-21 and described in Table 5-20.

Return to the Summary Table.

Source Transfer Step Register

**Figure 5-21. SRC_TRANSFER_STEP Register**

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

**Table 5-20. SRC_TRANSFER_STEP Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | SRCTRANSFERSTEP | R/W | 0h | These bits specify the change in the source address after a burst completes. The size must be a 16-bit two's complement value between -4096 and 4095 (inclusive). This value is added to the source address after each burst completes.<br>Reset type: SYSRSn<br>0h (R/W) = No address change<br>1h (R/W) = Add 1 to the address<br>2h (R/W) = Add 2 to the address<br>FFEh (R/W) = Add 4094 to the address<br>FFFh (R/W) = Add 4095 to the address<br>F000h (R/W) = Subtract 4096 from the address<br>F001h (R/W) = Subtract 4095 from the address<br>FFFEh (R/W) = Subtract 2 from the address<br>FFFFh (R/W) = Subtract 1 from the address |



Copyright © 2024 Texas Instruments Incorporated

### 5.9.3.10 DST_TRANSFER_STEP Register (Offset = 9h) [Reset = 0000h]

DST_TRANSFER_STEP is shown in Figure 5-22 and described in Table 5-21.

Return to the Summary Table.

Destination Transfer Step Register

**Figure 5-22. DST_TRANSFER_STEP Register**

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

**Table 5-21. DST_TRANSFER_STEP Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | DSTTRANSFERSTEP | R/W | 0h | These bits specify the change in the destination address after a burst completes. The size must be a 16-bit two's complement value between -4096 and 4095 (inclusive). This value is added to the destination address after each burst completes.<br>Reset type: SYSRSn<br>0h (R/W) = No address change<br>1h (R/W) = Add 1 to the address<br>2h (R/W) = Add 2 to the address<br>FFEh (R/W) = Add 4094 to the address<br>FFFh (R/W) = Add 4095 to the address<br>F000h (R/W) = Subtract 4096 from the address<br>F001h (R/W) = Subtract 4095 from the address<br>FFFEh (R/W) = Subtract 2 from the address<br>FFFFh (R/W) = Subtract 1 from the address |



Copyright © 2024 Texas Instruments Incorporated

### 5.9.3.11 SRC_WRAP_SIZE Register (Offset = Ah) [Reset = FFFFh]

SRC_WRAP_SIZE is shown in Figure 5-23 and described in Table 5-22.

Return to the Summary Table.

Source Wrap Size Register

**Figure 5-23. SRC_WRAP_SIZE Register**

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

**Table 5-22. SRC_WRAP_SIZE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | WRAPSIZE | R/W | FFFFh | These bits specify the number of bursts to transfer before the source address wraps around to the beginning address. The actual number is equal to WRAPSIZE + 1. To disable the wrapping function, set WRAPSIZE to a value larger than TRANSFERSIZE.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 5.9.3.12 SRC_WRAP_COUNT Register (Offset = Bh) [Reset = 0000h]

SRC_WRAP_COUNT is shown in Figure 5-24 and described in Table 5-23.

Return to the Summary Table.

Source Wrap Count Register

**Figure 5-24. SRC_WRAP_COUNT Register**

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

**Table 5-23. SRC_WRAP_COUNT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | WRAPSIZE | R | 0h | These bits indicate the number of bursts left before wrapping the source address.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 5.9.3.13 SRC_WRAP_STEP Register (Offset = Ch) [Reset = 0000h]

SRC_WRAP_STEP is shown in Figure 5-25 and described in Table 5-24.

Return to the Summary Table.

Source Wrap Step Register

**Figure 5-25. SRC_WRAP_STEP Register**

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

**Table 5-24. SRC_WRAP_STEP Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | WRAPSTEP | R/W | 0h | These bits specify the change in the source beginning address when the wrap counter reaches zero. The size must be a 16-bit two's complement value between -4096 and 4095 (inclusive). This value is added to the source address when wrapping occurs.<br>Reset type: SYSRSn<br>0h (R/W) = No address change<br>1h (R/W) = Add 1 to the address<br>2h (R/W) = Add 2 to the address<br>FFEh (R/W) = Add 4094 to the address<br>FFFh (R/W) = Add 4095 to the address<br>F000h (R/W) = Subtract 4096 from the address<br>F001h (R/W) = Subtract 4095 from the address<br>FFFEh (R/W) = Subtract 2 from the address<br>FFFFh (R/W) = Subtract 1 from the address |



Copyright © 2024 Texas Instruments Incorporated

### 5.9.3.14 DST_WRAP_SIZE Register (Offset = Dh) [Reset = FFFFh]

DST_WRAP_SIZE is shown in Figure 5-26 and described in Table 5-25.

Return to the Summary Table.

Destination Wrap Size Register

**Figure 5-26. DST_WRAP_SIZE Register**

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

**Table 5-25. DST_WRAP_SIZE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | WRAPSIZE | R/W | FFFFh | These bits specify the number of bursts to transfer before the destination address wraps around to the beginning address. The actual number is equal to WRAPSIZE + 1. To disable the wrapping function, set WRAPSIZE to a value larger than TRANSFERSIZE.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 5.9.3.15 DST_WRAP_COUNT Register (Offset = Eh) [Reset = 0000h]

DST_WRAP_COUNT is shown in Figure 5-27 and described in Table 5-26.

Return to the Summary Table.

Destination Wrap Count Register

**Figure 5-27. DST_WRAP_COUNT Register**

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

**Table 5-26. DST_WRAP_COUNT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | WRAPSIZE | R | 0h | These bits indicate the number of bursts left before wrapping the destination address.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 5.9.3.16 DST_WRAP_STEP Register (Offset = Fh) [Reset = 0000h]

DST_WRAP_STEP is shown in Figure 5-28 and described in Table 5-27.

Return to the Summary Table.

Destination Wrap Step Register

**Figure 5-28. DST_WRAP_STEP Register**

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

**Table 5-27. DST_WRAP_STEP Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | WRAPSTEP | R/W | 0h | These bits specify the change in the destination beginning address when the wrap counter reaches zero. The size must be a 16-bit two's complement value between -4096 and 4095 (inclusive). This value is added to the destination address when wrapping occurs.<br>Reset type: SYSRSn<br>0h (R/W) = No address change<br>1h (R/W) = Add 1 to the address<br>2h (R/W) = Add 2 to the address<br>FFEh (R/W) = Add 4094 to the address<br>FFFh (R/W) = Add 4095 to the address<br>F000h (R/W) = Subtract 4096 from the address<br>F001h (R/W) = Subtract 4095 from the address<br>FFFEh (R/W) = Subtract 2 from the address<br>FFFFh (R/W) = Subtract 1 from the address |



Copyright © 2024 Texas Instruments Incorporated

### 5.9.3.17 SRC_BEG_ADDR_SHADOW Register (Offset = 10h) [Reset = 00000000h]

SRC_BEG_ADDR_SHADOW is shown in Figure 5-29 and described in Table 5-28.

Return to the Summary Table.

Source Begin Address Shadow Register

**Figure 5-29. SRC_BEG_ADDR_SHADOW Register**

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

**Table 5-28. SRC_BEG_ADDR_SHADOW Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | BEGADDR | R/W | 0h | Shadow Source Beginning Address<br>At the start of a transfer, the value in this register is loaded into the SRC_BEG_ADDR_ACTIVE register and used as the beginning value for the source address. This register can be safely updated during a transfer.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 5.9.3.18 SRC_ADDR_SHADOW Register (Offset = 12h) [Reset = 00000000h]

SRC_ADDR_SHADOW is shown in Figure 5-30 and described in Table 5-29.

Return to the Summary Table.

Source Address Shadow Register

**Figure 5-30. SRC_ADDR_SHADOW Register**

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

**Table 5-29. SRC_ADDR_SHADOW Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | ADDR | R/W | 0h | Shadow Source Address<br>At the start of a transfer, the value in this register is loaded into the SRC_ADDR_ACTIVE register and used as the value of the source address. This register can be safely updated during a transfer.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 5.9.3.19 SRC_BEG_ADDR_ACTIVE Register (Offset = 14h) [Reset = 00000000h]

SRC_BEG_ADDR_ACTIVE is shown in Figure 5-31 and described in Table 5-30.

Return to the Summary Table.

Source Begin Address Active Register

**Figure 5-31. SRC_BEG_ADDR_ACTIVE Register**

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

**Table 5-30. SRC_BEG_ADDR_ACTIVE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | BEGADDR | R | 0h | Active Source Beginning Address<br>If a transfer is ongoing, this register holds the current beginning value for the source address. This address may be updated after wrapping.<br>When a transfer starts, this register is loaded with the shadow address from the SRC_BEG_ADDR_SHADOW register.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 5.9.3.20 SRC_ADDR_ACTIVE Register (Offset = 16h) [Reset = 00000000h]

SRC_ADDR_ACTIVE is shown in Figure 5-32 and described in Table 5-31.

Return to the Summary Table.

Source Address Active Register

**Figure 5-32. SRC_ADDR_ACTIVE Register**

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

**Table 5-31. SRC_ADDR_ACTIVE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | ADDR | R | 0h | Active Source Address<br>If a transfer is ongoing, this register holds the current value of the source address. This address may change after a write, a burst, or wrapping.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 5.9.3.21 DST_BEG_ADDR_SHADOW Register (Offset = 18h) [Reset = 00000000h]

DST_BEG_ADDR_SHADOW is shown in Figure 5-33 and described in Table 5-32.

Return to the Summary Table.

Destination Begin Address Shadow Register

**Figure 5-33. DST_BEG_ADDR_SHADOW Register**

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

**Table 5-32. DST_BEG_ADDR_SHADOW Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | BEGADDR | R/W | 0h | Shadow Destination Beginning Address<br>At the start of a transfer, the value in this register is loaded into the DST_BEG_ADDR_ACTIVE register and used as the beginning value for the destination address. This register can be safely updated during a transfer.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 5.9.3.22 DST_ADDR_SHADOW Register (Offset = 1Ah) [Reset = 00000000h]

DST_ADDR_SHADOW is shown in Figure 5-34 and described in Table 5-33.

Return to the Summary Table.

Destination Address Shadow Register

**Figure 5-34. DST_ADDR_SHADOW Register**

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

**Table 5-33. DST_ADDR_SHADOW Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | ADDR | R/W | 0h | Shadow Destination Address<br>At the start of a transfer, the value in this register is loaded into the DST_ADDR_ACTIVE register and used as the value of the destination address. This register can be safely updated during a transfer.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 5.9.3.23 DST_BEG_ADDR_ACTIVE Register (Offset = 1Ch) [Reset = 00000000h]

DST_BEG_ADDR_ACTIVE is shown in Figure 5-35 and described in Table 5-34.

Return to the Summary Table.

Destination Begin Address Active Register

**Figure 5-35. DST_BEG_ADDR_ACTIVE Register**

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

**Table 5-34. DST_BEG_ADDR_ACTIVE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | BEGADDR | R | 0h | Active Destination Beginning Address<br>If a transfer is ongoing, this register holds the current destination value for the source address. This address may be updated after wrapping.<br>When a transfer starts, this register is loaded with the shadow address from the DST_BEG_ADDR_SHADOW register.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

#### 5.9.3.24 DST_ADDR_ACTIVE Register (Offset = 1Eh) [Reset = 00000000h]

DST_ADDR_ACTIVE is shown in Figure 5-36 and described in Table 5-35.

Return to the Summary Table.

Destination Address Active Register

**Figure 5-36. DST_ADDR_ACTIVE Register**

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

**Table 5-35. DST_ADDR_ACTIVE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | ADDR | R | 0h | Active Destination Address<br>If a transfer is ongoing, this register holds the current value of the destination address. This address may change after a write, a burst, or wrapping.<br>Reset type: SYSRSn |

### *5.9.4 DMA Registers to Driverlib Functions*

**Table 5-36. DMA Registers to Driverlib Functions**

| File | Driverlib Function |
|---|---|
| **CTRL** | |
| dma.h | DMA_initController |
| **DEBUGCTRL** | |
| dma.h | DMA_setEmulationMode |
| **PRIORITYCTRL1** | |
| dma.h | DMA_setPriorityMode |
| **PRIORITYSTAT** | |
| - | |
| **MODE** | |
| dma.c | DMA_configMode |
| dma.h | DMA_enableTrigger |
| dma.h | DMA_disableTrigger |
| dma.h | DMA_enableInterrupt |
| dma.h | DMA_disableInterrupt |
| dma.h | DMA_enableOverrunInterrupt |
| dma.h | DMA_disableOverrunInterrupt |
| dma.h | DMA_setInterruptMode |
| **CONTROL** | |
| dma.h | DMA_triggerSoftReset |
| dma.h | DMA_forceTrigger |
| dma.h | DMA_clearTriggerFlag |
| dma.h | DMA_getTransferStatusFlag |
| dma.h | DMA_getBurstStatusFlag |
| dma.h | DMA_getRunStatusFlag |
| dma.h | DMA_getOverflowFlag |
| dma.h | DMA_getTriggerFlagStatus |
| dma.h | DMA_startChannel |



Copyright © 2024 Texas Instruments Incorporated

**Table 5-36. DMA Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| dma.h | DMA_stopChannel |
| dma.h | DMA_clearErrorFlag |
| **BURST_SIZE** | |
| dma.c | DMA_configBurst |
| **BURST_COUNT** | |
| - | |
| **SRC_BURST_STEP** | |
| dma.c | DMA_configBurst |
| **DST_BURST_STEP** | |
| dma.c | DMA_configBurst |
| **TRANSFER_SIZE** | |
| dma.c | DMA_configTransfer |
| **TRANSFER_COUNT** | |
| - | |
| **SRC_TRANSFER_STEP** | |
| dma.c | DMA_configTransfer |
| **DST_TRANSFER_STEP** | |
| dma.c | DMA_configTransfer |
| **SRC_WRAP_SIZE** | |
| dma.c | DMA_configWrap |
| **SRC_WRAP_COUNT** | |
| - | |
| **SRC_WRAP_STEP** | |
| dma.c | DMA_configWrap |
| **DST_WRAP_SIZE** | |
| dma.c | DMA_configWrap |
| **DST_WRAP_COUNT** | |
| - | |
| **DST_WRAP_STEP** | |
| dma.c | DMA_configWrap |
| **SRC_BEG_ADDR_SHADOW** | |
| dma.c | DMA_configAddresses |
| dma.h | DMA_configSourceAddress |
| **SRC_ADDR_SHADOW** | |
| dma.c | DMA_configAddresses |
| dma.h | DMA_configSourceAddress |
| **SRC_BEG_ADDR_ACTIVE** | |
| - | |
| **SRC_ADDR_ACTIVE** | |
| - | |
| **DST_BEG_ADDR_SHADOW** | |
| dma.c | DMA_configAddresses |
| dma.h | DMA_configDestAddress |
| **DST_ADDR_SHADOW** | |
| dma.c | DMA_configAddresses |



Copyright © 2024 Texas Instruments Incorporated

**Table 5-36. DMA Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| dma.h | DMA_configDestAddress |
| **DST_BEG_ADDR_ACTIVE** | |
| - | |
| **DST_ADDR_ACTIVE** | |
| - | |



Copyright © 2024 Texas Instruments Incorporated

Chapter 6

# Control Law Accelerator (CLA)

The Control Law Accelerator (CLA) Type-1 is an independent, fully-programmable, 32-bit floating-point math processor that brings concurrent control-loop execution to the C28x family. The low interrupt latency of the CLA allows the CLA to read ADC samples "just-in-time." This significantly reduces the ADC sample to output delay to enable faster system response and higher MHz control loops. By using the CLA to service time-critical control loops, the main CPU is free to perform other system tasks such as communications and diagnostics. This chapter provides an overview of the architectural structure and components of the control law accelerator.


**6.1 Introduction**....718
**6.2 CLA Interface**....720
**6.3 CLA and CPU Arbitration**....726
**6.4 CLA Configuration and Debug**....729
**6.5 Pipeline**....732
**6.6 Software**....738
**6.7 Instruction Set**....739
**6.8 CLA Registers**....870





Copyright © 2024 Texas Instruments Incorporated

## 6.1 Introduction

The Control Law Accelerator extends the capabilities of the C28x CPU by adding parallel processing. Time-critical control loops serviced by the CLA can achieve low ADC sample to output delay. Thus, the CLA enables faster system response and higher frequency control loops. Utilizing the CLA for time-critical tasks frees up the main CPU to perform other system and communication functions concurrently.

### *6.1.1 Features*

The following is a list of major features of the CLA:

- C compilers are available for CLA software development.
- Clocked at the same rate as the main CPU (SYSCLKOUT).
- An independent architecture allowing CLA algorithm execution independent of the main C28x CPU.
  - Complete bus architecture:
    - Program Address Bus (PAB) and Program Data Bus (PDB)
    - Data Read Address Bus (DRAB), Data Read Data Bus (DRDB), Data Write Address Bus (DWAB), and Data Write Data Bus (DWDB)
  - Independent eight stage pipeline.
  - 16-bit program counter (MPC)
  - Four 32-bit result registers (MR0-MR3)
  - Two 16-bit auxiliary registers (MAR0, MAR1)
  - Status register (MSTF)
- Instruction set includes:
  - IEEE single-precision (32-bit) floating-point math operations
  - Floating-point math with parallel load or store
  - Floating-point multiply with parallel add or subtract
  - 1/X and 1/sqrt(X) estimations
  - Data type conversions.
  - Conditional branch and call
  - Data load/store operations
- The CLA program code can consist of up to eight tasks or interrupt service routines
  - The start address of each task is specified by the MVECT registers.
  - No limit on task size as long as the tasks fit within the configurable CLA program memory space.
  - One task is serviced at a time until completion. There is no nesting of tasks.
  - Upon task completion a task-specific interrupt is flagged within the PIE.
  - When a task finishes the next highest-priority pending task is automatically started.
- Task trigger mechanisms:
  - C28x CPU using the IACK instruction
  - Task1 to Task8: up to 256 possible trigger sources from peripherals connected to the shared bus on which the CLA assumes secondary ownership.
- Memory and Shared Peripherals:
  - Two dedicated message RAMs for communication between the CLA and the main CPU.
  - The C28x CPU can map CLA program and data memory to the main CPU space or CLA space.
  - The CLA, on reset, is the secondary master for all peripherals that can have either the CLA or DMA as their secondary master.

### *6.1.2 CLA Related Collateral*

**Foundational Materials**

- C2000 Academy - CLA
- C2000 CLA C Compiler Series (Video)
- CLA Hands On Workshop (Video)
- CLA usage in Valley Switching Boost Power Factor Correction (PFC) Reference Design (Video)



Copyright © 2024 Texas Instruments Incorporated

- *Enhancing the Computational Performance of the C2000™ Microcontroller Family Application Report*

**Getting Started Materials**

- CLA Software Development Guide
- *Software Examples to Showcase Unique Capabilities of TI's C2000™ CLA Application Report*

**Expert Materials**

- *Digital Control of Two Phase Interleaved PFC and Motor Drive Using MCU With CLA Application Report*
- *Sensorless Field Oriented Control:3-Phase Perm.Magnet Synch. Motors With CLA Application Report*

### *6.1.3 Block Diagram*

Figure 6-1 is a block diagram of the CLA.

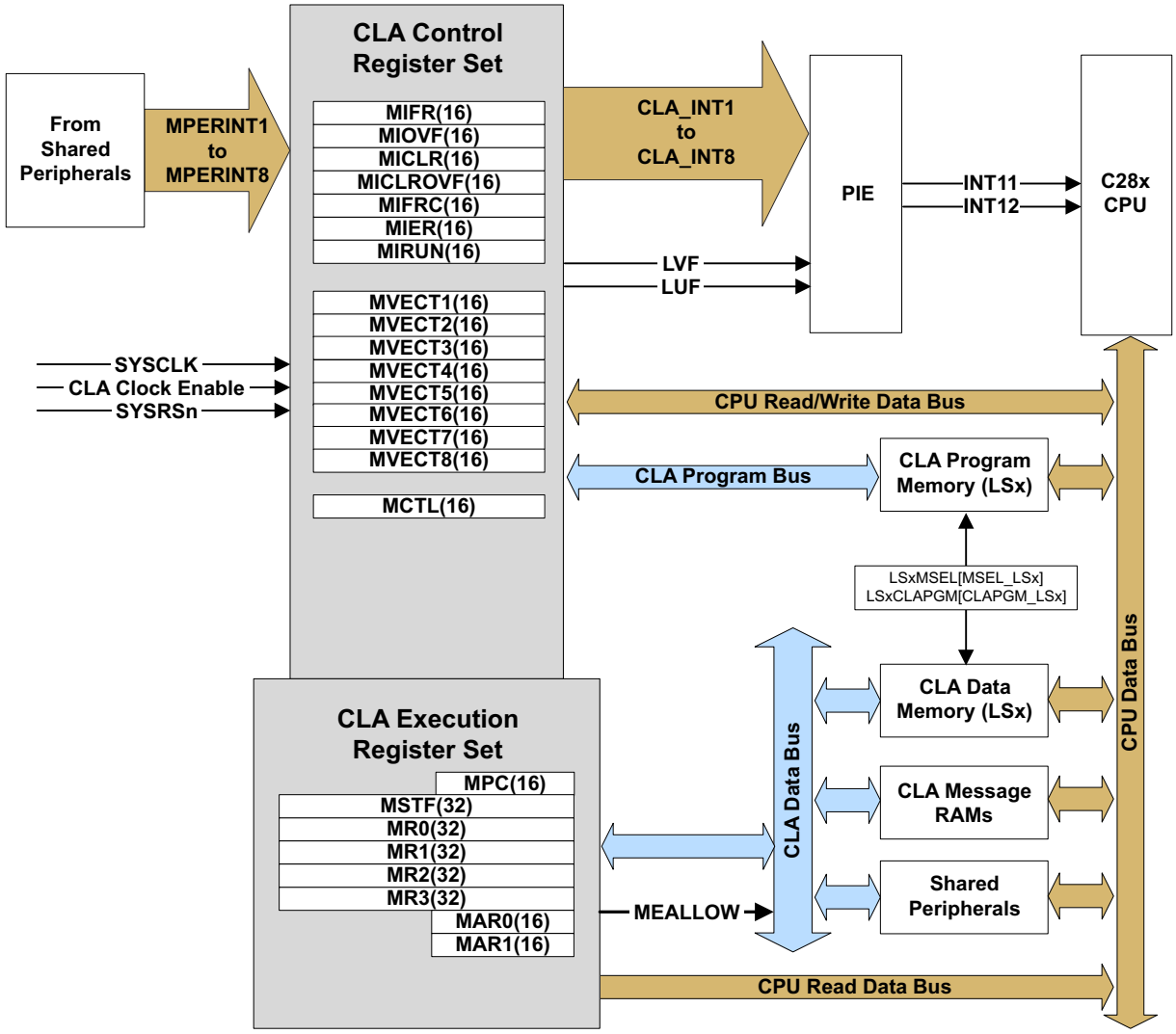



**Figure 6-1. CLA Block Diagram**



Copyright © 2024 Texas Instruments Incorporated

## 6.2 CLA Interface

This section describes how the C28x main CPU can interface to the CLA and conversely.

### *6.2.1 CLA Memory*

The CLA can access three types of memory: program, data and message RAMs. The behavior and arbitration for each type of memory is described in this chapter. The CLA RAMs are protected by the DCSM module. Refer to the *Dual Code Security Module (DCSM)* section of the *System Control and Interrupts* chapter for more details on the security scheme.

- **CLA Program Memory**

  The CLA program can be loaded with any of the local shared memories (LSxRAM) on the core. At reset, all memory blocks are mapped to the CPU. While mapped to the CPU space, the CPU can copy the CLA program code into the memory. During debug, the memory can also be loaded directly by the Code Composer Studio™ IDE.

  Once the memory is initialized with CLA code, the CPU maps the memory to the CLA program space by:
  1. Assigning ownership of the memory block to the CLA by writing a 1 to the memory block's MemCfgRegs.LSxMSEL[MSEL_LSx] bit.
  2. Specifying the memory block as a code block for the CLA by writing a 1 to the MemCfgRegs.LSxCLAPGM[CLAPGM_LSx] bit.

  When a memory block is configured as CLA program memory, debug accesses are allowed only on cycles where the CLA is not fetching a new instruction. A detailed explanation of the memory configurations and access arbitration (CPU, CLA, and DEBUG) process can be found in the *Memory Controller Module* section of the *System Control and Interrupts* chapter.

  All CLA program fetches are performed as 32-bit read operations and all opcodes must be aligned to an even address. Since all CLA opcodes are 32-bits, this alignment occurs naturally.

- **CLA Data Memory**

  Any of the device's LSxRAMs can serve as data memory blocks to the CLA. At reset, all blocks are mapped to the CPU memory space, whereby the CPU can initialize the memory with data tables, coefficients, and so on, for the CLA to use.

  Once the memory is initialized with CLA data, the CPU maps the memory to the CLA data space by:
  1. Assigning ownership of the memory block to the CLA by writing a 1 to the memory block's MemCfgRegs.LSxMSEL[MSEL_LSx] bit.
  2. Specifying the memory block as a data block for the CLA by writing a 0 to the MemCfgRegs.LSxCLAPGM[CLAPGM_LSx] bit. The value of this bit at reset is 0.

  When a memory block is configured as CLA data memory, CLA read and write accesses are arbitrated along with CPU accesses. The user has the option of turning on CPU fetch or write protection to the memory by writing to the appropriate bits of the MemCfgRegs.LSxACCPROTx registers. A detailed explanation of the memory configurations and access arbitration (CPU, CLA, and DEBUG) process can be found in the *Memory Controller Module* section of the *System Control and Interrupts* chapter.



Copyright © 2024 Texas Instruments Incorporated

- **CLA Shared Message RAMs**

  There are two memory blocks for data sharing and communication between the CLA and the CPU on each CPU subsystem. The message RAMs are always mapped to both CPU and CLA memory spaces, and only data access is allowed; no program fetches can be performed.
  - **CLA to CPU Message RAM:** The CLA can use this block to pass data to the CPU. This block is both readable and writable by the CLA. This block is also readable by the CPU but writes by the CPU are ignored.
  - **CPU to CLA Message RAM:** The CPU can use this block to pass data and messages to the CLA. This message RAM is both readable and writable by the CPU. The CLA can perform reads but writes by the CLA are ignored.

### *6.2.2 CLA Memory Bus*

The CLA has dedicated bus architecture similar to that of the C28x CPU where there are separate program read, data read, and data write buses. Thus, there can be simultaneous instruction fetch, data read, and data write in a single cycle. Like the C28x CPU, the CLA expects memory logic to align any 32-bit read or write to an even address. If the address-generation logic generates an odd address, the CLA can begin reading or writing at the previous even address. This alignment does not affect the address values generated by the address-generation logic.

- **CLA Program Bus**

  The CLA program bus has an access range of 32K 32-bit instructions. Since all CLA instructions are 32 bits, this bus always fetches 32 bits at a time and the opcodes must be even-word aligned. The amount of program space available for the CLA is limited to the number of available LSxRAM blocks. This number is device-dependent and can be described in the data sheet.

- **CLA Data Read Bus**

  The CLA data read bus has a 64K x 16 address range. The bus can perform 16 or 32-bit reads and can automatically stall if there are memory access conflicts. The data read bus has access to both the message RAMs, CLA data memory, and the shared peripherals.

- **CLA Data Write Bus**

  The CLA data write bus has a 64K x 16 address range. This bus can perform 16 or 32-bit writes. The bus can automatically stall if there are memory access conflicts. The data write bus has access to the CLA to CPU message RAM, CLA data memory, and the shared peripherals.



Copyright © 2024 Texas Instruments Incorporated

### 6.2.3 Shared Peripherals and EALLOW Protection

For a given CPU subsystem, the CLA and DMA share secondary access to some peripherals. The secondary ownership of the bus is determined by the CpuSysRegs.SECMSEL[VBUS32_x] bit. If the bit is set to 0, the CLA is the secondary owner. If the bit is set to 1, the DMA is the secondary owner. By default, at reset, the CLA is given the secondary ownership of the bus and, therefore, can access all the peripherals connected to the bus.

**Note**

The CLA read access time to the bus is 2-wait states while write access is 0-wait.

Refer to the device data sheet for the list of peripherals connected to the bus.

Several peripheral control registers are protected from spurious 28x CPU writes by the EALLOW protection mechanism. These same registers are also protected from spurious CLA writes. The EALLOW bit in the CPU status register 1 (ST1) indicates the state of protection for the CPU. Likewise, the MEALLOW bit in the CLA status register (MSTF) indicates the state of write protection for the CLA. The MEALLOW CLA instruction enables write access by the CLA to EALLOW protected registers. Likewise, the MEDIS CLA instruction disables write access. This way the CLA can enable and disable write access independent of the CPU.

The ADC offers the option to generate an early interrupt pulse at the start of a sample conversion. If this option is used to start an ADC-triggered CLA task, use the intervening cycles until the completion of the conversion to perform preliminary calculations or loads and stores before finally reading the ADC value. The CLA pipeline activity for this scenario is shown in Section 6.5.

### 6.2.4 CLA Tasks and Interrupt Vectors

The CLA program code is divided up into tasks or interrupt service routines. Tasks do not have a fixed starting location or length. The CLA program memory can be divided up as desired. The CLA uses the contents of the interrupt vectors (MVECT1 to MVECT8) to determine where a task begins; tasks are terminated by the MSTOP instruction.

The CLA supports eight tasks. Task 1 has the highest priority and task 8 has the lowest priority.

A task can be requested by a peripheral interrupt or by software:

- **Peripheral interrupt trigger**

  Each task can be triggered by software-selectable interrupt sources. The trigger for each task is defined by writing an appropriate value to the DmaClaSrcSelRegs.CLA1TASKSRCSELx[TASKx] bit field. Each option specifies an interrupt source from a specific peripheral on the shared bus. The peripheral interrupt triggers are listed in Table 6-1.

  For example, task 1 (MVECT1) can be set to trigger on EPWMINT1 by writing 36 to DmaClaSrcSelRegs.CLA1TASKSRCSEL1.TASK1. To disable the triggering of a task by a peripheral, set the DmaClaSrcSelRegs.CLA1TASKSRCSELx[TASKx] bit field to 0. Note that a CLA task only triggers on a level transition (an edge) of the configured interrupt source.



Copyright © 2024 Texas Instruments Incorporated

**Table 6-1. Configuration Options**

| Select Value | CLA Trigger Source |
|---|---|
| 0 | CLA_SOFTWARE_TRIGGER |
| 1 | ADCAINT1 |
| 2 | ADCAINT2 |
| 3 | ADCAINT3 |
| 4 | ADCAINT4 |
| 5 | ADCA_EVT_INT |
| 6 | ADCBINT1 |
| 7 | ADCBINT2 |
| 8 | ADCBINT3 |
| 9 | ADCBINT4 |
| 10 | ADCB_EVT_INT |
| 11 | ADCCINT1 |
| 12 | ADCCINT2 |
| 13 | ADCCINT3 |
| 14 | ADCCINT4 |
| 15 | ADCC_EVT_INT |
| 16 | ADCDINT1 |
| 17 | ADCDINT2 |
| 18 | ADCDINT3 |
| 19 | ADCDINT4 |
| 20 | ADCD_EVT_INT |
| 21-28 | Reserved |
| 29 | XINT1 |
| 30 | XINT2 |
| 31 | XINT3 |
| 32 | XINT4 |
| 33 | XINT5 |
| 34-35 | Reserved |
| 36 | EPWM1_INT |
| 37 | EPWM2_INT |
| 38 | EPWM3_INT |
| 39 | EPWM4_INT |
| 40 | EPWM5_INT |
| 41 | EPWM6_INT |
| 42 | EPWM7_INT |
| 43 | EPWM8_INT |
| 44 | EPWM9_INT |
| 45 | EPWM10_INT |



Copyright © 2024 Texas Instruments Incorporated

**Table 6-1. Configuration Options (continued)**

| Select Value | CLA Trigger Source |
|---|---|
| 46 | EPWM11_INT |
| 47 | EPWM12_INT |
| 48-67 | Reserved |
| 68 | CPU_TINT0 |
| 69 | CPU_TINT1 |
| 70 | CPU_TINT2 |
| 71 | MCBSPA_TX |
| 72 | MCBSPA_RX |
| 73 | MCBSPB_TX |
| 74 | MCBSPB_RX |
| 75 | ECAP1_INT |
| 76 | ECAP2_INT |
| 77 | ECAP3_INT |
| 78 | ECAP4_INT |
| 79 | ECAP5_INT |
| 80 | ECAP6_INT |
| 81-82 | Reserved |
| 83 | EQEP1_INT |
| 84 | EQEP2_INT |
| 85 | EQEP3_INT |
| 86-94 | Reserved |
| 95 | SD1_ERRINT |
| 96 | SD2_ERRINT |
| 97-106 | Reserved |
| 107 | UPPA_INT |
| 108 | Reserved |
| 109 | SPIA_TXINT |
| 110 | SPIA_RXINT |
| 111 | SPIB_TXINT |
| 112 | SPIB_RXINT |
| 113 | SPIC_TXINT |
| 114 | SPIC_RXINT |
| 115-126 | Reserved |
| 127 | CLB1_INT |
| 128 | CLB2_INT |
| 129 | CLB3_INT |
| 130 | CLB4_INT |
| 131-255 | Reserved |



Copyright © 2024 Texas Instruments Incorporated

- **Software Trigger**

  CPU software can trigger tasks by writing to the MIFRC register or by the IACK instruction. Using the IACK instruction is more efficient because the instruction does not require the need to issue an EALLOW to set MIFR bits. Set the MCTL[IACKE] bit to enable the IACK feature. Each bit in the operand of the IACK instruction corresponds to a task. For example, IACK #0x0001 sets bit 0 in the MIFR register to start task 1. Likewise, IACK #0x0003 set bits 0 and 1 in the MIFR register to start task 1 and task 2.

The CLA has a fetch mechanism and can run and execute a task independent of the CPU. Only one task is serviced at a time; there is no nesting of tasks. The task currently running is indicated in the MIRUN register. Interrupts that have been received but not yet serviced are indicated in the flag register (MIFR). If an interrupt request from a peripheral is received and that same task is already flagged, then the overflow flag bit is set. Overflow flags remain set until the flags are cleared by the CPU.

If the CLA is idle (no task is currently running), then the highest priority interrupt request that is both flagged (MIFR) and enabled (MIER) starts.

The flow is as follows:

1. The associated RUN register bit is set (MIRUN) and the flag bit (MIFR) is cleared.
2. The CLA begins execution at the location indicated by the associated interrupt vector (MVECTx). MVECT contains the absolute 16-bit address of the task in the lower 64K memory space.
3. The CLA executes instructions until the MSTOP instruction is found. This indicates the end of the task.
4. The MIRUN bit is cleared.
5. The task-specific interrupt to the PIE is issued. This informs the main CPU that the task has completed.
6. The CLA returns to idle.

Once a task completes the next highest-priority pending task is automatically serviced and this sequence repeats.

### *6.2.5 CLA Software Interrupt to CPU*

The CLA can issue a software interrupt to the C28x CPU (on the same subsystem) at any point in the code through the use of the CLA1SOFTINTEN and CLA1INTFRC registers. See Section 6.8 for a description of these registers. If a software interrupt is selected for a CLA task, then an end-of-task interrupt is not issued to the C28x CPU when that task completes.



Copyright © 2024 Texas Instruments Incorporated

## 6.3 CLA and CPU Arbitration

Typically, CLA activity is independent of the CPU activity. Under the circumstance where the CLA or CPU attempt to concurrently access memory or a peripheral register within the same interface, an arbitration procedure occurs. This section describes this arbitration.

The arbitration follows a fixed arbitration scheme with highest priority first:

1. CLA WRITE
2. CLA READ
3. CPU WRITE
4. CPU READ

Refer to the Memory Controller Module section of the *System Control and Interrupts* chapter.

### *6.3.1 CLA Message RAM*

Message RAMs consist of two blocks:

- CLA to CPU Message RAM
- CPU to CLA Message RAM

These blocks are useful for passing data between the CLA and CPU. No opcode fetches, from either the CLA or CPU, are allowed from the message RAMs. A write protection violation is not generated if the CLA attempts to write to the CPU to CLA message RAM, but the write is ignored. The arbitration scheme for the message RAMs are the same as those for the shared memories, described in the Memory Controller Module section of the *System Control and Interrupts* chapter.

The message RAMs have the following characteristics:

- CLA to CPU Message RAM:
  The following accesses are allowed:
  - CPU reads
  - CLA data reads and writes
  - CPU debug reads and writes

  The following accesses are ignored:
  - CPU writes
- CPU to CLA Message RAM:
  The following accesses are allowed:
  - CPU reads and writes
  - CLA reads
  - CPU debug reads and writes

  The following accesses are ignored:
  - CLA writes



Copyright © 2024 Texas Instruments Incorporated

### 6.3.2 CLA Program Memory

The behavior of the program memory depends on the state of the MMEMCFG[PROGE] bit. This bit controls whether the memory is mapped to CLA space or CPU space.

- **MMEMCFG[PROGE] == 0**

  In this case, the memory is mapped to the CPU. The CLA is halted and no tasks can be incoming.
  - Any CLA fetch is treated as an illegal opcode condition as described in Section 6.4.4. This condition does not occur, if the proper procedure is followed to map the program memory.
  - CLA reads and writes cannot occur
  - The memory block behaves as any normal RAM block mapped to CPU memory space.

  Priority of accesses are (highest priority first):
  1. CPU data write, program write, debug write
  2. CPU data read, program read, debug read
  3. CPU fetch, program read

- **MMEMCFG[PROGE] == 1**

  In this case, the memory block is mapped to CLA space. The CPU can only make debug accesses.
  - CLA reads and writes cannot occur
  - CLA fetches are allowed
  - CPU fetches return 0 that is an illegal opcode and causes an ITRAP interrupt.
  - CPU data reads and program reads return 0
  - CPU data writes and program writes are ignored

  Priority of accesses are (highest priority first):
  1. CLA fetch
  2. CPU debug write
  3. CPU debug read

**Note**

Because the CLA fetch has higher priority than CPU debug reads, there is a possibility for the CLA to permanently block debug accesses if the CLA is executing in a loop. This can occur when initially developing CLA code due to a bug. To avoid this issue, the program memory returns all 0x0000 for CPU debug reads (ignore writes) when the CLA is running. When the CLA is halted or idle, then normal CPU debug read and write access can be performed.



Copyright © 2024 Texas Instruments Incorporated

### 6.3.3 CLA Data Memory

There are independent data memory blocks. The behavior of the data memory depends on the state of the MMEMCFG[RAM0E] MMEMCFG[RAM1E] bits. These bits determine whether the memory blocks are mapped to CLA space or CPU space.

- **MMEMCFG[RAMxE] == 0**

  In this case the memory block is mapped to the CPU.
  - CLA fetches cannot occur to this block.
  - CLA reads return 0.
  - CLA writes are ignored.
  - The memory block behaves as any normal RAM block mapped to the CPU memory space.

  Priority of accesses are (highest priority first):
  1. CPU data write/program write/debug access write
  2. CPU data read/debug access read
  3. CPU fetch/program read

- **MMEMCFG[RAMxE] == 1**

  In this case the memory block is mapped to CLA space. The CPU can make only debug accesses.
  - CLA fetches cannot occur to this block.
  - CLA read and CLA writes are allowed.
  - CPU fetches return 0
  - CPU data reads and program reads return 0.
  - CPU data writes and program writes are ignored.

  Priority of accesses are (highest priority first):
  1. CLA data write
  2. CPU debug write
  3. CPU debug read
  4. CLA read

### 6.3.4 Peripheral Registers (ePWM, HRPWM, Comparator)

Accesses to the registers follow these rules:

- If both the CPU and CLA request access at the same time, then the CLA has priority and the main CPU is stalled.
- If a CPU access is in-progress and another CPU access is pending, then the CLA has priority over the pending CPU access. In this case, the CLA access begins when the current CPU access completes.
- While a CPU access is in-progress, any incoming CLA access is stalled.
- While a CLA access is in-progress, any incoming CPU access is stalled.
- A CPU write operation has priority over a CPU read operation.
- A CLA write operation has priority over a CLA read operation.
- If the CPU is performing a read-modify-write operation and the CLA performs a write to the same location, the CLA write can be lost if the operation occurs in-between the CPU read and write. For this reason, do not mix CPU and CLA accesses to same location.



Copyright © 2024 Texas Instruments Incorporated

## 6.4 CLA Configuration and Debug

This section discusses the steps necessary to configure and debug the CLA.

### *6.4.1 Building a CLA Application*

The control law accelerator can be programmed in either CLA assembly code, using the instructions described in Section 6.7, or a reduced subset of the C language. CLA assembly code resides in the same project with C28x code. The only restriction is the CLA code must be in the assembly section. This can be easily done using the **.sect** assembly directive. This does not prevent CLA and C28x code from being linked into the same memory region in the linker command file.

System and CLA initialization are performed by the main CPU. This can typically be done in C or C++ but can also include C28x assembly code. The main CPU also copies the CLA code to the program memory and, if needed, initialize the CLA data RAMs. Once system initialization is complete and the application begins, the CLA services the interrupts using the CLA assembly code (or tasks). The main CPU can perform other tasks concurrently with CLA program execution.

The CLA Type 1 requires Codegen V6.2.4 or later with the compiler switch: --cla_support=cla1.

### *6.4.2 Typical CLA Initialization Sequence*

A typical CLA initialization sequence is performed by the main CPU as described in this section.

1. **Copy CLA code into the CLA program RAM:** The source for the CLA code can initially reside in the Flash or a data stream from a communications peripheral or anywhere the main CPU can access. The debugger can also be used to load code directly to the CLA program RAM during development.
2. **Initialize CLA data RAM, if necessary:** Populate the CLA data RAM with any required data coefficients or constants.
3. **Configure the CLA registers:** Configure the CLA registers, but keep interrupts disabled until later (leave MIER = 0):
   - **Enable the CLA peripheral clock using the assigned PCLKCRn register:** The peripheral clock control (PCLKCRn) registers are defined in the *System Control and Interrupts* chapter.
   - **Populate the CLA task interrupt vectors:**
     - MVECT1 to MVECT8

       Each vector needs to be initialized with the start address of the task to be executed when the CLA receives the associated interrupt. This address is the full 16-bit starting address of the task in the lower 64K section of memory.
   - **Select the task interrupt sources:** For each task select the interrupt source in the CLA1TASKSRCSELx register. If a task is software triggered, select no interrupt.
   - **Enable IACK to start a task from software, if desired:** To enable the IACK instruction to start a task set the MCTL[IACKE] bit. Using the IACK instruction avoids having to set and clear the EALLOW bit.
   - **Map CLA data RAM to CLA space, if necessary:** Map the data RAM to the CLA space by first, assigning ownership of the memory block to the CLA by writing a 1 to the memory block's MemCfgRegs.LSxMSEL[MSEL_LSx] bit, and then specifying the memory block as a CLA data block by writing a 0 to the MemCfgRegs.LSxCLAPGM[CLAPGM_LSx] bit. When an LSx memory is configured as a CLA data memory, the CLA read/write accesses are arbitrated along with CPU accesses. The user has the option of turning on CPU fetch or write protection to the memory by writing to the appropriate bits of the MemCfgRegs.LSxACCPROTx registers.
   - **Map CLA program RAM to CLA space:** Map the CLA program RAM to CLA space by first assigning ownership of the memory block to the CLA by writing a 1 to the memory block's MemCfgRegs.LSxMSEL[MSEL_LSx] bit, and then specifying the memory block as CLA code memory by writing a 1 to the MemCfgRegs.LSxCLAPGM[CLAPGM_LSx] bit. When an LSx memory is configured as CLA program memory, only debug accesses are allowed on cycles in which the CLA is not fetching a new instruction.
4. **Initialize the PIE vector table and registers:** When a CLA task completes, the associated interrupt in the PIE is flagged. The CLA overflow and underflow flags also have associated interrupts within the PIE.



Copyright © 2024 Texas Instruments Incorporated

5. **Enable CLA tasks/interrupts:** Set appropriate bits in the interrupt enable register (MIER) to allow the CLA to service interrupts. Note that a CLA task only triggers on a level transition (a falling edge) of the configured interrupt source. If a peripheral is enabled and an interrupt fires before the CLA is configured, then the CLA does not recognize the interrupt edge and does not respond. To avoid this, configure the CLA before the peripherals or clear any pending peripheral interrupts before setting bits in the MIER register.
6. **Initialize other peripherals:** Initialize any peripherals (such as ePWM, ADC, and others) that generate interrupt triggers for enabled CLA tasks.

   The CLA is now ready to service interrupts and the message RAMs can be used to pass data between the CPU and the CLA. Mapping of the CLA program and data RAMs typically occurs only during the initialization process. If the RAM mapping needs to be changed after initialization, the CLA interrupts must be disabled and all tasks must be completed (by checking the MIRUN register) prior to modifying the RAM ownership.

### *6.4.3 Debugging CLA Code*

Debugging the CLA code is a simple process that occurs independently of the main CPU.

#### 6.4.3.1 Breakpoint Support (MDEBUGSTOP)

1. **Insert a breakpoint in CLA code**

   Insert a CLA breakpoint (MDEBUGSTOP instruction) into the code where the CLA is to halt, then rebuild and reload the code. Because the CLA does not flush the pipeline when in single-step, the MDEBUGSTOP instruction must be inserted as part of the code. The debugger cannot insert the MDEBUGSTOP instruction as needed.

   If CLA breakpoints are not enabled, then the MDEBUGSTOP instruction is ignored and is treated as a MNOP. The MDEBUGSTOP instruction can be placed anywhere in the CLA code as long as the MDEBUGSTOP instruction is not within three instructions of a MBCNDD, MCCNDD, or MRCNDD instruction. When programming in C, the user can use the __mdebugstop() intrinsic instead; the compiler makes sure that the placement of the MDEBUSTOP instruction in the generated assembly does not violate any of the pipeline restrictions.
2. **Enable CLA breakpoints**

   Enable the CLA breakpoints in the debugger. In the Code Composer Studio™ IDE, this is done by connecting to the CLA core (or tap) from the debug perspective. Breakpoints are disabled when the core is disconnected.
3. **Start the task**

   There are three ways to start the task:
   a. The peripheral can assert an interrupt,
   b. The main CPU can execute an IACK instruction, or
   c. The user can manually write to the MIFRC register in the debugger window

   When the task starts, the CLA executes instructions until the MDEBUGSTOP is in the D2 phase of the pipeline. At this point, the CLA halts and the pipeline is frozen. The MPC register reflects the address of the MDEBUGSTOP instruction.



Copyright © 2024 Texas Instruments Incorporated

4. **Single-step the CLA code**

   Once halted, the user can single-step the CLA code. The behavior of a CLA single-step is different than the main C28x. When issuing a CLA single-step, the pipeline is clocked only one cycle and then again frozen. On the C28x CPU, the pipeline is flushed for each single-step.

   Run to the next MDEBUGSTOP or to the end of the task. If another task is pending, the task automatically starts when run to the end of the task.

   **Note**

   A CLA fetch has higher priority than CPU debug reads. For this reason, the CLA to permanently block CPU debug accesses if the CLA is executing in a loop is possible. This can occur when initially developing CLA code due to a bug that causes an infinite loop. To avoid locking up the main CPU, the program memory returns all 0x0000 for CPU debug reads when the CLA is running. When the CLA is halted or idle, then normal CPU debug read and write access to CLA program memory can be performed.

   If the CLA gets caught in an infinite loop, use a soft or hard reset to exit the condition. A debugger reset also exits the condition.

   There are special cases that can occur when single-stepping a task such that the program counter, MPC, reaches the MSTOP instruction at the end of the task.

   - **MPC halts at or after the MSTOP with a task already pending**

     If single-stepping or halted in "task A" and "task B" comes in before the MPC reaches the MSTOP, then "task B" starts if continuing to step through the MSTOP instruction. Basically, if "task B" is pending before the MPC reaches MSTOP in "task A" then there is no issue in "task B" starting and no special action is required.
   - **MPC halts at or after the MSTOP with no task pending**

     In this case, if single-stepped or halted in "task A" and the MPC has reached the MSTOP with no tasks pending. If "task B" comes in at this point, "task B" is flagged in the MIFR register but "task B" can or cannot start if continuing to single-step through the MSTOP instruction of "task A."

     Depending on exactly when the new task comes in, to reliably start "task B", perform a soft reset and reconfigure the MIER bits. Once this is done, start single-stepping "task B."

     This case can be handled slightly differently if there is control over when "task B" comes in (for example, using the IACK instruction to start the task). In this case, the task is single-stepped or halted in "task A" and the MPC has reached the MSTOP with no tasks pending. Before forcing "task B," run free to force the CLA out of the debug state. Once this is done, force "task B" and continue debugging.

5. **Disable CLA breakpoints, if desired**

   In the Code Composer Studio™ IDE, disable the CLA breakpoints by disconnecting the CLA core in the debug perspective. Make sure to first issue a run or reset; otherwise, the CLA is halted and no other tasks start.

### *6.4.4 CLA Illegal Opcode Behavior*

If the CLA fetches an opcode that does not correspond to a legal instruction, the CLA behaves as follows:

- The CLA halts with the illegal opcode in the D2 phase of the pipeline as if a breakpoint. This occurs whether CLA breakpoints are enabled or not.
- The CLA issues the task-specific interrupt to the PIE.
- The MIRUN bit for the task remains set.

Further single-stepping is ignored once execution halts due to an illegal op-code. To exit this situation, issue either a soft or hard reset of the CLA as described in Section 6.4.5.



Copyright © 2024 Texas Instruments Incorporated

### 6.4.5 Resetting the CLA

There are times when resetting the CLA is needed. For example, during code debug the CLA can enter an infinite loop due to a code bug. The CLA has two types of resets: hard and soft. Both of these resets can be performed by the debugger or by the main CPU.

- **Hard Reset** Writing a 1 to the MCTL[HARDRESET] bit performs a hard reset of the CLA. The behavior of a hard reset is the same as a system reset (using $\overline{\text{XRS}}$ or the debugger). In this case, all CLA configuration and execution registers can be set to the default state and CLA execution halts.
- **Soft Reset** Writing a 1 to the MCTL[SOFTRESET] bit performs a soft reset of the CLA. If a task is executing, the task halts and the associated MIRUN bit is cleared. All bits within the interrupt enable (MIER) register are also cleared, so that no new tasks start.

## 6.5 Pipeline

This section describes the CLA pipeline stages and presents cases where pipeline alignment must be considered.

### 6.5.1 Pipeline Overview

The CLA pipeline is very similar to the C28x pipeline with eight stages:

1. **Fetch 1 (F1):** During the F1 stage the program read address is placed on the CLA program address bus.
2. **Fetch 2 (F2):** During the F2 stage the instruction is read using the CLA program data bus.
3. **Decode 1 (D1):** During D1 the instruction is decoded.
4. **Decode 2 (D2):** Generate the data read address. Changes to MAR0 and MAR1 due to post-increment using indirect addressing takes place in the D2 phase. Conditional branch decisions are also made at this stage based on the MSTF register flags.
5. **Read 1 (R1):** Place the data read address on the CLA data-read address bus. If a memory conflict exists, the R1 stage is stalled.
6. **Read 2 (R2):** Read the data value using the CLA data read data bus.
7. **Execute (EXE):** Execute the operation. Changes to MAR0 and MAR1 due to loading an immediate value or value from memory take place in this stage.
8. **Write (W):** Place the write address and write data on the CLA write data bus. If a memory conflict exists, the W stage is stalled.



Copyright © 2024 Texas Instruments Incorporated

### 6.5.2 CLA Pipeline Alignment

The majority of the CLA instructions do not require any special pipeline considerations. This section lists the few operations that do require special consideration.

- **Write Followed by Read**

  In both the C28x pipeline and the CLA pipeline, the read operation occurs before the write. This means that if a read operation immediately follows a write, then the read completes first as shown in Table 6-2. In most cases this does not cause a problem since the contents of one memory location does not depend on the state of another. For accesses to peripherals where a write to one location can affect the value in another location, the code must wait for the write to complete before issuing the read as shown in Table 6-3.

  This behavior is different for the C28x CPU. For the C28x CPU, any write followed by read to the same location is protected by what is called write-followed-by-read protection. This protection automatically stalls the pipeline so that the write completes before the read. In addition, some peripheral frames are protected such that a C28x CPU write to one location within the frame always completes before a read to the frame. The CLA does not have this protection mechanism. Instead, the code must wait to perform the read.

**Table 6-2. Write Followed by Read - Read Occurs First**

| Instruction | F1 | F2 | D1 | D2 | R1 | R2 | E | W |
|---|---|---|---|---|---|---|---|---|
| I1 MMOV16 @Reg1, MR3 | I1 | | | | | | | |
| I2 MMOV16 MR2, @Reg2 | I2 | I1 | | | | | | |
| | | I2 | I1 | | | | | |
| | | | I2 | I1 | | | | |
| | | | | I2 | I1 | | | |
| | | | | | I2 | I1 | | |
| | | | | | | I2 | I1 | |
| | | | | | | | I2 | I1 |

**Table 6-3. Write Followed by Read - Write Occurs First**

| Instruction | F1 | F2 | D1 | D2 | R1 | R2 | E | W |
|---|---|---|---|---|---|---|---|---|
| I1 MMOV16 @Reg1, MR3 | I1 | | | | | | | |
| I2 | I2 | I1 | | | | | | |
| I3 | I3 | I2 | I1 | | | | | |
| I4 | I4 | I3 | I2 | I1 | | | | |
| I5 MMOV16 MR2, @Reg2 | I5 | I4 | I3 | I2 | I1 | | | |
| | | I5 | I4 | I3 | I2 | I1 | | |
| | | | I5 | I4 | I3 | I2 | I1 | |
| | | | | I5 | I4 | I3 | I2 | I1 |
| | | | | | I5 | I4 | I3 | |
| | | | | | | I5 | I4 | |
| | | | | | | | I5 | |



Copyright © 2024 Texas Instruments Incorporated

- **Delayed Conditional instructions: MBCNDD, MCCNDD, and MRCNDD**

  Referring to Example 6-1, the following applies to delayed conditional instructions:
  - **I1:** I1 is the last instruction that can effect the CNDF flags for the branch, call, or return instruction. The CNDF flags are tested in the D2 phase of the pipeline. That is, a decision is made whether to branch or not when MBCNDD, MCCNDD, or MRCNDD is in the D2 phase.
  - **I2, I3, and I4:** The three instructions preceding MBCNDD can change the MSTF flags but have no effect on whether the MBCNDD instruction branches or not. This is because the flag modification occurs after the D2 phase of the branch, call, or return instruction. These three instructions must not be a MSTOP, MDEBUGSTOP, MBCNDD, MCCNDD, or MRCNDD.
  - **I5, I6, and I7:** The three instructions following a branch, call, or return are always executed irrespective of whether the condition is true or not. These instructions must not be MSTOP, MDEBUGSTOP, MBCNDD, MCCNDD, or MRCNDD.

  For a more detailed description, refer to the description for MBCNDD, MCCNDD, and MRCNDD.

***Example 6-1. Code Fragment For MBCNDD, MCCNDD, or MRCNDD***

```
  <Instruction 1>    ; I1 Last instruction that can affect flags for
                     ;     the branch, call or return operation
  <Instruction 2>    ; I2 Cannot be stop, branch, call or return
  <Instruction 3>    ; I3 Cannot be stop, branch, call or return
  <Instruction 4>    ; I4 Cannot be stop, branch, call or return
  <branch/call/ret>  ; MBCNDD, MCCNDD or MRCNDD
                     ; I5-I7: Three instructions after are always
                     ; executed whether the branch/call or return is
                     ; taken or not
  <Instruction 5>    ; I5 Cannot be stop, branch, call or return
  <Instruction 6>    ; I6 Cannot be stop, branch, call or return
  <Instruction 7>    ; I7 Cannot be stop, branch, call or return
  <Instruction 8>    ; I8
  <Instruction 9>    ; I9
  ....
```

- **Stop or Halting a Task: MSTOP and MDEBUGSTOP**

  The MSTOP and MDEBUGSTOP instructions cannot be placed three instructions before or after a conditional branch, call or return instruction (MBCNDD, MCCNDD, or MRCNDD). Refer to Example 6-1. To single-step through a branch/call or return, insert the MDEBUGSTOP at least four instructions back and step from there.



Copyright © 2024 Texas Instruments Incorporated

- **Loading MAR0 or MAR1**

  A load of auxiliary register MAR0 or MAR1 occurs in the EXE phase of the pipeline. Any post increment of MAR0 or MAR1 using indirect addressing occurs in the D2 phase of the pipeline. Referring to Example 6-2, the following applies when loading the auxiliary registers:
  - **I1 and I2:** The two instructions following the load instruction use the value in MAR0 or MAR1 before the update occurs.
  - **I3:** Loading of an auxiliary register occurs in the EXE phase while updates due to post-increment addressing occur in the D2 phase. Thus I3 cannot use the auxiliary register or there is a conflict. In the case of a conflict, the update due to address-mode post increment wins and the auxiliary register is not updated with #_X.
  - **I4:** Starting with the 4th instruction MAR0 or MAR1 has the new value.

***Example 6-2. Code Fragment for Loading MAR0 or MAR1***

```
; Assume MAR0 is 50 and #_X is 20

   MMOVI16 MAR0, #_X ; Load MAR0 with address of X (20)
   <Instruction 1>   ; I1 uses the old value of MAR0 (50)
   <Instruction 2>   ; I2 uses the old value of MAR0 (50)
   <Instruction 3>   ; I3 Cannot use MAR0
   <Instruction 4>   ; I4 uses the new value of MAR0 (20)
   <Instruction 5>   ; I5 uses the new value of MAR0 (20)
   ....
```



Copyright © 2024 Texas Instruments Incorporated

#### 6.5.2.1 ADC Early Interrupt to CLA Response

The ADC can be configured to generate an early interrupt pulse before the ADC conversion completes. If this option is used to start a CLA task, the CLA is able to read the result as soon as the conversion result is available in the ADC result register. This combination of just-in-time sampling along with the low interrupt response of the CLA enable faster system response and higher frequency control loops. The CLA task trigger to first instruction fetch interrupt latency is 4 cycles.

Timings for ADC conversions are shown in the timing diagrams of the ADC chapter. If the ADCCLK is a divided down version of the SYSCLK, the user has to account for the conversion time in SYSCLK cycles.

For example, if using the 12-bit ADC with ADCCLK at SYSCLK / 4, the ADC can take 10.5 ADCCLK x 4 SYSCLK = 42 SYSCLK cycles to complete a conversion. If using the ADC in 16-bit mode at the same ADCCLK, the ADC can take 29.5 ADCCLK x 4 SYSCLK = 118 SYSCLK cycles, and so on.

From a CLA perspective, the pipeline activity is shown in Table 6-4 for an N-cycle (SYSCLK) ADC conversion. The N-2 instruction arrives in the R2 phase just in time to read the result register. While the prior instructions enter the R2 phase of the pipeline too soon to read the conversion, the instructions can be efficiently used for pre-processing calculations needed by the task.

**Table 6-4. ADC to CLA Early Interrupt Response**

| ADC Activity | CLA Activity | F1 | F2 | D1 | D2 | R1 | R2 | E | W |
|---|---|---|---|---|---|---|---|---|---|
| Sample | | | | | | | | | |
| Sample | | | | | | | | | |
| ... | | | | | | | | | |
| Sample | | | | | | | | | |
| $Conversion_{(Cycle\ 1)}$ | Interrupt Received | | | | | | | | |
| $Conversion_{(Cycle\ 2)}$ | Task Startup | | | | | | | | |
| $Conversion_{(Cycle\ 3)}$ | Task Startup | | | | | | | | |
| $Conversion_{(Cycle\ 4)}$ | $I_{(Cycle\ 4)}$ | $I_{(Cycle\ 4)}$ | | | | | | | |
| $Conversion_{(Cycle\ 5)}$ | $I_{(Cycle\ 5)}$ | $I_{(Cycle\ 5)}$ | $I_{(Cycle\ 4)}$ | | | | | | |
| $Conversion_{(...)}$ | ... | ... | ... | ... | ... | ... | ... | | |
| $Conversion_{(Cycle\ N-6)}$ | $I_{(Cycle\ N-6)}$ | $I_{(Cycle\ N-6)}$ | $I_{(Cycle\ N-7)}$ | $I_{(Cycle\ N-8)}$ | $I_{(Cycle\ N-9)}$ | $I_{(Cycle\ N-10)}$ | $I_{(Cycle\ N-11)}$ | | |
| $Conversion_{(Cycle\ N-5)}$ | $I_{(Cycle\ N-5)}$ | $I_{(Cycle\ N-5)}$ | $I_{(Cycle\ N-6)}$ | $I_{(Cycle\ N-7)}$ | $I_{(Cycle\ N-8)}$ | $I_{(Cycle\ N-9)}$ | $I_{(Cycle\ N-10)}$ | | |
| $Conversion_{(Cycle\ N-4)}$ | $I_{(Cycle\ N-4)}$ | $I_{(Cycle\ N-4)}$ | $I_{(Cycle\ N-5)}$ | $I_{(Cycle\ N-6)}$ | $I_{(Cycle\ N-7)}$ | $I_{(Cycle\ N-8)}$ | $I_{(Cycle\ N-9)}$ | | |
| $Conversion_{(Cycle\ N-3)}$ | $I_{(Cycle\ N-3)}$ | $I_{(Cycle\ N-3)}$ | $I_{(Cycle\ N-4)}$ | $I_{(Cycle\ N-5)}$ | $I_{(Cycle\ N-6)}$ | $I_{(Cycle\ N-7)}$ | $I_{(Cycle\ N-8)}$ | | |
| $Conversion_{(Cycle\ N-2)}$ | **Read RESULT** | **Read RESULT** | $I_{(Cycle\ N-3)}$ | $I_{(Cycle\ N-4)}$ | $I_{(Cycle\ N-5)}$ | $I_{(Cycle\ N-6)}$ | $I_{(Cycle\ N-7)}$ | | |
| $Conversion_{(Cycle\ N-1)}$ | | | **Read RESULT** | $I_{(Cycle\ N-3)}$ | $I_{(Cycle\ N-4)}$ | $I_{(Cycle\ N-5)}$ | $I_{(Cycle\ N-6)}$ | | |
| $Conversion_{(Cycle\ N-0)}$ | | | | **Read RESULT** | $I_{(Cycle\ N-3)}$ | $I_{(Cycle\ N-4)}$ | $I_{(Cycle\ N-5)}$ | | |
| Conversion Complete | | | | | **Read RESULT** | $I_{(Cycle\ N-3)}$ | $I_{(Cycle\ N-4)}$ | | |
| RESULT Latched | | | | | | **Read RESULT** | $I_{(Cycle\ N-3)}$ | | |
| **RESULT Available** | | | | | | | **Read RESULT** | | |



Copyright © 2024 Texas Instruments Incorporated

### 6.5.3 Parallel Instructions

Parallel instructions are single opcodes that perform two operations in parallel. The following types of parallel instructions are available: math operation in parallel with a move operation, or two math operations in parallel. Both operations complete in a single cycle and there are no special pipeline alignment requirements.

**Example 6-3. Math Operation with Parallel Load**

```
;  MADDF32 || MMOV32 instruction: 32-bit floating-point add with parallel move
;  MADDF32 is a 1 cycle operation
;  MMOV32 is a 1 cycle operation
         MADDF32    MR0,  MR1, #2     ; MR0 = MR1 + 2,
      || MMOV32   MR1,  @Val          ; MR1 gets the contents of Val
                                      ; <-- MMOV32 completes here (MR1 is valid)
                                      ; <-- DDF32 completes here (MR0 is valid)
         MMPYF32 MR0, MR0, MR1        ; Any instruction, can use MR1 and/or MR0
```

**Example 6-4. Multiply with Parallel Add**

```
;  MMPYF32 || MADDF32 instruction: 32-bit floating-point multiply with parallel add
;  MMPYF32 is a 1 cycle operation
;  MADDF32 is a 1 cycle operation
     MMPYF32 MR0, MR1, MR3         ; MR0 = MR1 * MR3
 ||  MADDF32 MR1, MR2, MR0         ; MR1 = MR2 + MR0 (Uses value of MR0 before MMPYF32)
                                   ; <-- MMPYF32 and MADDF32 complete here (MR0 and MR1 are valid)
     MMPYF32 MR1, MR1, MR0         ; Any instruction, can use MR1 and/or MR0
```

### 6.5.4 CLA Task Execution Latency

The CLA task execution latency depends on the state of the system:

- CLA task trigger of new task (normal or background) without background task active:

  Task takes 8 cycles from CLA task trigger to first instruction of task to reach the D2 phase of pipeline.

  **Note**

  If background task has been configured in the system, then the compiler during code compilation adds context save instructions at the start of each regular task and restore instructions at end of each task so that register content can be saved and restored in case a background task is executing while the regular task is triggered. When a regular task is entered, this compiler-generated context save instruction is the first instruction of the task.

- CLA task trigger of normal task when background task is active:

  Task takes 9 cycles from CLA task trigger to first instruction of normal task to reach the D2 phase of pipeline. There is a difference of one clock cycle to force the MSTOP in the D2 phase of the background task before the task exits as compared to a new task trigger without the background task active.

  **Note**

  If the MBCNDD/MCCNDD/MRCNDD instructions in the background task are in the D2 phase of the pipeline when a new task gets triggered, the task takes a minimum of 3 more cycles to complete these uninterruptible instructions adding to the delay.

- Returning to background task from normal task:

  The task takes 5 cycles to return from a normal task to resume the background task instruction at the D2 phase of the pipeline.



Copyright © 2024 Texas Instruments Incorporated

## 6.6 Software

### *6.6.1 CLA Examples*

NOTE: These examples are located in the C2000Ware installation at the following location:
C2000Ware_VERSION#/driverlib/DEVICE_GPN/examples/CORE_IF_MULTICORE/cla

Cloud access to these examples is available at the following link: dev.ti.com C2000Ware Examples.

#### 6.6.1.1 CLA arcsine(x) using a lookup table (cla_asin_cpu01)

FILE: cla_ex1_asin.c

In this example, Task 1 of the CLA will calculate the arcsine of an input argument in the range (-1.0 to 1.0) using a lookup table.

*Memory Allocation*

- CLA1 Math Tables (RAMLS0)
- CLA1 to CPU Message RAM
  - fResult - Result of the lookup algorithm
- CPU to CLA1 Message RAM
  - fVal - Sample input to the lookup algorithm

*Watch Variables*

- fVal - Argument to task 1
- fResult - Result of arcsin(fVal)

#### 6.6.1.2 CLA arctangent(x) using a lookup table (cla_atan_cpu01)

FILE: cla_ex2_atan.c

In this example, Task 1 of the CLA will calculate the arctangent of an input argument using a lookup table.

*Memory Allocation*

- CLA1 Math Tables (RAMLS0)
- CLA1 to CPU Message RAM
  - fResult - Result of the lookup algorithm
- CPU to CLA1 Message RAM
  - fNum - Numerator of sample input
  - fDen - Denominator of sample input

*Watch Variables*

- fVal - Argument to task 1
- fResult - Result of arctan(fVal)



Copyright © 2024 Texas Instruments Incorporated

## 6.7 Instruction Set

This section describes the assembly language instructions of the control law accelerator. Also described are parallel operations, conditional operations, resource constraints, and addressing modes. The instructions listed here are independent from C28x and C28x+FPU instruction sets.

### *6.7.1 Instruction Descriptions*

This section gives detailed information on the instruction set. Each instruction presents the following information:

- Operands
- Opcode
- Description
- Exceptions
- Pipeline
- Examples
- See also

The example INSTRUCTION is shown to familiarize you with the way each instruction is described. The example describes the kind of information you find in each part of the individual instruction description and where to obtain more information. CLA instructions follow the same format as the C28x instructions; the source operands are always on the right and the destination operands are on the left.

The explanations for the syntax of the operands used in the instruction descriptions for the CLA are given in Table 6-5.

**Table 6-5. Operand Nomenclature**

| Symbol | Description |
|---|---|
| #16FHi | 16-bit immediate (hex or float) value that represents the upper 16-bits of an IEEE 32-bit floating-point value. Lower 16-bits of the mantissa are assumed to be zero. |
| #16FHiHex | 16-bit immediate hex value that represents the upper 16-bits of an IEEE 32-bit floating-point value. Lower 16-bits of the mantissa are assumed to be zero. |
| #16FLoHex | A 16-bit immediate hex value that represents the lower 16-bits of an IEEE 32-bit floating-point value |
| #32Fhex | 32-bit immediate value that represents an IEEE 32-bit floating-point value |
| #32F | Immediate float value represented in floating-point representation |
| #0.0 | Immediate zero |
| #SHIFT | Immediate value of 1 to 32 used for arithmetic and logical shifts. |
| addr | Opcode field indicating the addressing mode |
| CNDF | Condition to test the flags in the MSTF register |
| FLAG | Selected flags from MSTF register (OR) 8 bit mask indicating which floating-point status flags to change |
| MAR0 | Auxiliary register 0 |
| MAR1 | Auxiliary register 1 |
| MARx | Either MAR0 or MAR1 |
| mem16 | 16-bit memory location accessed using direct, indirect, or offset addressing modes |
| mem32 | 32-bit memory location accessed using direct, indirect, or offset addressing modes |
| MRa | MR0 to MR3 registers |
| MRb | MR0 to MR3 registers |
| MRc | MR0 to MR3 registers |
| MRd | MR0 to MR3 registers |
| MRe | MR0 to MR3 registers |
| MRf | MR0 to MR3 registers |
| MSTF | CLA Floating-point Status Register |
| shift | Opcode field indicating the number of bits to shift. |
| VALUE | Flag value of 0 or 1 for selected flag (OR) 8 bit mask indicating the flag value; 0 or 1 |



Copyright © 2024 Texas Instruments Incorporated

Each instruction has a table that gives a list of the operands and a short description. Instructions always have the destination operands first followed by the source operands.

**Table 6-6. INSTRUCTION dest, source1, source2 Short Description**

| | Description |
|---|---|
| dest1 | Description for the 1st operand for the instruction |
| source1 | Description for the 2nd operand for the instruction |
| source2 | Description for the 3rd operand for the instruction |
| Opcode | This section shows the opcode for the instruction |
| Description | Detailed description of the instruction execution is described. Any constraints on the operands imposed by the processor or the assembler are discussed. |
| Restrictions | Any constraints on the operands or use of the instruction imposed by the processor are discussed. |
| Pipeline | This section describes the instruction in terms of pipeline cycles as described in Section 6.5 |
| Example | Examples of instruction execution. If applicable, register and memory values are given before and after instruction execution. Some examples are code fragments while other examples are full tasks that assume the CLA is correctly configured and the main CPU has passed the CLA data. |
| Operands | Each instruction has a table that gives a list of the operands and a short description. Instructions always have the destination operands first followed by the source operands. |

### *6.7.2 Addressing Modes and Encoding*

The CLA uses the same address to access data and registers as the main CPU. For example, if the main CPU accesses an ePWM register at address 0x00 6800, then the CLA accesses the register using address 0x6800. Since all CLA accessible memory and registers are within the low 64k x 16 of memory, only the low 16-bits of the address are used by the CLA.

To address the CLA data memory, message RAMs and shared peripherals, the CLA supports two addressing modes:

- Direct addressing mode: Uses the address of the variable or register directly.
- Indirect addressing with 16-bit post increment. This mode uses either XAR0 or XAR1.

The CLA does not use a data page pointer or a stack pointer. The two addressing modes are encoded as shown Table 6-7.



Copyright © 2024 Texas Instruments Incorporated

**Table 6-7. Addressing Modes**

| Addressing Mode | 'addr' Opcode Field Encode(1) | Description |
|---|---|---|
| **@dir** | **0000** | **Direct Addressing Mode**<br>Example 1: MMOV32 MR1, @_VarA<br>Example 2: MMOV32 MR1, @_EPwm1Regs.CMPA.all<br>In this case, the 'mmmm mmmm mmmm mmmm' opcode field is populated with the 16-bit address of the variable. This is the low 16-bits of the address to access the variable using the main CPU.<br>For example, @_VarA populates the address of the variable VarA. and @_EPwm1Regs.CMPA.all populates the address of the CMPA register. |
| ***MAR0[#imm16]++** | **0001** | **MAR0 Indirect Addressing with 16-bit Immediate Post Increment** |
| ***MAR1[#imm16]++** | **0010** | **MAR1 Indirect Addressing with 16-bit Immediate Post Increment**<br>addr = MAR0 (or MAR1) Access memory using the address stored in MAR0 (or MAR1).<br>MAR0 (or MAR1) += #imm16 Then post increment MAR0 (or MAR1) by #imm16.<br>Example 1: MMOV32 MR0, *MAR0[2]++<br>Example 2: MMOV32 MR1, *MAR1[-2]++<br>For a post increment of 0, the assembler accepts both *MAR0 and *MAR0[0]++.<br>The 'mmmm mmmm mmmm mmmm' opcode field is populated with the signed 16-bit pointer offset. For example, if #imm16 is 2, then the opcode field is 0x0002. Likewise, if #imm16 is -2, then the opcode field is 0xFFFE.<br>If addition of the 16-bit immediate causes overflow, then the value wraps around on a 16-bit boundary. |
| ***MAR0+[#imm16]** | **0101** | **MAR0 Offset Addressing with 16-bit Immediate Offset** |
| ***MAR1+[#imm16]** | **0110** | **MAR1 Offset Addressing with 16-bit Immediate Offset**<br>addr = MAR0 (or MAR1) + #imm16to the base Add the offset #imm16 address stored in MAR0(MAR1) to access the desired memory location<br>Example 1: MMOV32 MR0, *MAR0+[2]<br>Example 1: MMOV32 MR1, *MAR1+[-2]<br>The ‘mmmm mmmm mmmm mmmm’ opcode field is populated with the signed 16-bit pointer offset. For example, if #imm16 is 2, then the opcode field is 0x0002. Likewise, if #imm16 is -2, then the opcode field is 0xFFFE.<br>If the addition of the 16-bit immediate causes overflow, the value wraps around on a 16-bit boundary. |

(1) Values not shown are reserved.

Encoding for the shift fields in the MASR32, MLSR32 and MLSL32 instructions is shown in Table 6-8.

**Table 6-8. Shift Field Encoding**

| Shift Value | 'shift' Opcode Field Encode |
|---|---|
| 1 | 0000 |
| 2 | 0001 |
| 3 | 0010 |
| .... | .... |
| 32 | 1111 |



Copyright © 2024 Texas Instruments Incorporated

For instructions that use MRx (where x can be 'a' through 'f') as operands, the trailing alphabet appears in the opcode as a two-bit field. For example:

```
MMPYF32 MRa, MRb, MRc ||
MADDF32 MRd, MRe, MRf
```

whose opcode is,

```
LSW: 0000 ffee ddcc bbaa
MSW: 0111 1010 0000 0000
```

The two-bit field specifies one of four working registers according to Table 6-9.

**Table 6-9. Operand Encoding**

| Two-Bit Field | Working Register |
|---|---|
| 00 | MR0 |
| 01 | MR1 |
| 10 | MR2 |
| 11 | MR3 |

Table 6-10 shows the condition field encoding for conditional instructions such as MNEGF, MSWAPF, MBCNDD, MCCNDD, and MRCNDD.

**Table 6-10. Condition Field Encoding**

| Encode(1) | CNDF | Description | MSTF Flags Tested |
|---|---|---|---|
| 0000 | NEQ | Not equal to zero | ZF == 0 |
| 0001 | EQ | Equal to zero | ZF == 1 |
| 0010 | GT | Greater than zero | ZF == 0 AND NF == 0 |
| 0011 | GEQ | Greater than or equal to zero | NF == 0 |
| 0100 | LT | Less than zero | NF == 1 |
| 0101 | LEQ | Less than or equal to zero | ZF == 1 OR NF == 1 |
| 1010 | TF | Test flag set | TF == 1 |
| 1011 | NTF | Test flag not set | TF == 0 |
| 1100 | LU | Latched underflow | LUF == 1 |
| 1101 | LV | Latched overflow | LVF == 1 |
| 1110 | UNC | Unconditional | None |
| 1111 | UNCF(2) | Unconditional with flag modification | None |

(1) Values not shown are reserved.
(2) This is the default operation, if no CNDF field is specified. This condition allows the ZF and NF flags to be modified when a conditional operation is executed. All other conditions do not modify these flags.



Copyright © 2024 Texas Instruments Incorporated

### 6.7.3 Instructions

The instructions are listed alphabetically.

**Instruction Set Summary**

**MABSF32 MRa, MRb** — 32-Bit Floating-Point Absolute Value....745
**MADD32 MRa, MRb, MRc** — 32-Bit Integer Add....746
**MADDF32 MRa, #16FHi, MRb** — 32-Bit Floating-Point Addition....747
**MADDF32 MRa, MRb, #16FHi** — 32-Bit Floating-Point Addition....749
**MADDF32 MRa, MRb, MRc** — 32-Bit Floating-Point Addition....751
**MADDF32 MRd, MRe, MRf||MMOV32 mem32, MRa** — 32-Bit Floating-Point Addition with Parallel Move....752
**MADDF32 MRd, MRe, MRf ||MMOV32 MRa, mem32** — 32-Bit Floating-Point Addition with Parallel Move....753
**MAND32 MRa, MRb, MRc** — Bitwise AND....755
**MASR32 MRa, #SHIFT** — Arithmetic Shift Right....756
**MBCNDD 16BitDest {, CNDF}** — Branch Conditional Delayed....758
**MCCNDD 16BitDest {, CNDF}** — Call Conditional Delayed....763
**MCMP32 MRa, MRb** — 32-Bit Integer Compare for Equal, Less Than or Greater Than....767
**MCMPF32 MRa, MRb** — 32-Bit Floating-Point Compare for Equal, Less Than or Greater Than....769
**MCMPF32 MRa, #16FHi** — 32-Bit Floating-Point Compare for Equal, Less Than or Greater Than....770
**MDEBUGSTOP** — Debug Stop Task....772
**MEALLOW** — Enable CLA Write Access to EALLOW Protected Registers....773
**MEDIS** — Disable CLA Write Access to EALLOW Protected Registers....774
**MEINVF32 MRa, MRb** — 32-Bit Floating-Point Reciprocal Approximation....775
**MEISQRTF32 MRa, MRb** — 32-Bit Floating-Point Square-Root Reciprocal Approximation....777
**MF32TOI16 MRa, MRb** — Convert 32-Bit Floating-Point Value to 16-Bit Integer....779
**MF32TOI16R MRa, MRb** — Convert 32-Bit Floating-Point Value to 16-Bit Integer and Round....780
**MF32TOI32 MRa, MRb** — Convert 32-Bit Floating-Point Value to 32-Bit Integer....781
**MF32TOUI16 MRa, MRb** — Convert 32-Bit Floating-Point Value to 16-bit Unsigned Integer....783
**MF32TOUI16R MRa, MRb** — Convert 32-Bit Floating-Point Value to 16-bit Unsigned Integer and Round....784
**MF32TOUI32 MRa, MRb** — Convert 32-Bit Floating-Point Value to 32-Bit Unsigned Integer....785
**MFRACF32 MRa, MRb** — Fractional Portion of a 32-Bit Floating-Point Value....786
**MI16TOF32 MRa, MRb** — Convert 16-Bit Integer to 32-Bit Floating-Point Value....787
**MI16TOF32 MRa, mem16** — Convert 16-Bit Integer to 32-Bit Floating-Point Value....788
**MI32TOF32 MRa, mem32** — Convert 32-Bit Integer to 32-Bit Floating-Point Value....789
**MI32TOF32 MRa, MRb** — Convert 32-Bit Integer to 32-Bit Floating-Point Value....790
**MLSL32 MRa, #SHIFT** — Logical Shift Left....791
**MLSR32 MRa, #SHIFT** — Logical Shift Right....793
**MMACF32 MR3, MR2, MRd, MRe, MRf ||MMOV32 MRa, mem32** — 32-Bit Floating-Point Multiply and Accumulate with Parallel Move....794
**MMAXF32 MRa, MRb** — 32-Bit Floating-Point Maximum....797
**MMAXF32 MRa, #16FHi** — 32-Bit Floating-Point Maximum....799
**MMINF32 MRa, MRb** — 32-Bit Floating-Point Minimum....801
**MMINF32 MRa, #16FHi** — 32-Bit Floating-Point Minimum....803
**MMOV16 MARx, MRa, #16I** — Load the Auxiliary Register with MRa + 16-bit Immediate Value....805
**MMOV16 MARx, mem16** — Load MAR1 with 16-bit Value....808
**MMOV16 mem16, MARx** — Move 16-Bit Auxiliary Register Contents to Memory....811
**MMOV16 mem16, MRa** — Move 16-Bit Floating-Point Register Contents to Memory....812
**MMOV32 mem32, MRa** — Move 32-Bit Floating-Point Register Contents to Memory....814
**MMOV32 mem32, MSTF** — Move 32-Bit MSTF Register to Memory....816
**MMOV32 MRa, mem32 {, CNDF}** — Conditional 32-Bit Move....817
**MMOV32 MRa, MRb {, CNDF}** — Conditional 32-Bit Move....819
**MMOV32 MSTF, mem32** — Move 32-Bit Value from Memory to the MSTF Register....821
**MMOVD32 MRa, mem32** — Move 32-Bit Value from Memory with Data Copy....822
**MMOVF32 MRa, #32F** — Load the 32-Bits of a 32-Bit Floating-Point Register....824



Copyright © 2024 Texas Instruments Incorporated

**MMOVI16 MARx, #16I** — Load the Auxiliary Register with the 16-Bit Immediate Value..... 826
**MMOVI32 MRa, #32FHex** — Load the 32-Bits of a 32-Bit Floating-Point Register with the Immediate..... 828
**MMOVIZ MRa, #16FHi** — Load the Upper 16-Bits of a 32-Bit Floating-Point Register .....830
**MMOVZ16 MRa, mem16** — Load MRx with 16-Bit Value.....831
**MMOVXI MRa, #16FLoHex** — Move Immediate Value to the Lower 16-Bits of a Floating-Point Register.....832
**MMPYF32 MRa, MRb, MRc** — 32-Bit Floating-Point Multiply.....833
**MMPYF32 MRa, #16FHi, MRb** — 32-Bit Floating-Point Multiply .....834
**MMPYF32 MRa, MRb, #16FHi** — 32-Bit Floating-Point Multiply .....836
**MMPYF32 MRa, MRb, MRc||MADDF32 MRd, MRe, MRf** — 32-Bit Floating-Point Multiply with Parallel Add...838
**MMPYF32 MRd, MRe, MRf ||MMOV32 MRa, mem32** — 32-Bit Floating-Point Multiply with Parallel Move.....840
**MMPYF32 MRd, MRe, MRf ||MMOV32 mem32, MRa** — 32-Bit Floating-Point Multiply with Parallel Move.....842
**MMPYF32 MRa, MRb, MRc ||MSUBF32 MRd, MRe, MRf** — 32-Bit Floating-Point Multiply with Parallel Subtract .....843
**MNEGF32 MRa, MRb{, CNDF}** — Conditional Negation.....845
**MNOP** — No Operation.....847
**MOR32 MRa, MRb, MRc** — Bitwise OR.....848
**MRCNDD {CNDF}** — Return Conditional Delayed.....849
**MSETFLG FLAG, VALUE** — Set or Clear Selected Floating-Point Status Flags.....852
**MSTOP** — Stop Task.....853
**MSUB32 MRa, MRb, MRc** — 32-Bit Integer Subtraction.....855
**MSUBF32 MRa, MRb, MRc** — 32-Bit Floating-Point Subtraction.....856
**MSUBF32 MRa, #16FHi, MRb** — 32-Bit Floating-Point Subtraction.....857
**MSUBF32 MRd, MRe, MRf ||MMOV32 MRa, mem32** — 32-Bit Floating-Point Subtraction with Parallel Move.... 859
**MSUBF32 MRd, MRe, MRf ||MMOV32 mem32, MRa** — 32-Bit Floating-Point Subtraction with Parallel Move.... 860
**MSWAPF MRa, MRb {, CNDF}** — Conditional Swap..... 861
**MTESTTF CNDF** — Test MSTF Register Flag Condition.....863
**MUI16TOF32 MRa, mem16** — Convert Unsigned 16-Bit Integer to 32-Bit Floating-Point Value.....865
**MUI16TOF32 MRa, MRb** — Convert Unsigned 16-Bit Integer to 32-Bit Floating-Point Value..... 866
**MUI32TOF32 MRa, mem32** — Convert Unsigned 32-Bit Integer to 32-Bit Floating-Point Value.....867
**MUI32TOF32 MRa, MRb** — Convert Unsigned 32-Bit Integer to 32-Bit Floating-Point Value..... 868
**MXOR32 MRa, MRb, MRc** — Bitwise Exclusive Or.....869



Copyright © 2024 Texas Instruments Incorporated

## MABSF32 MRa, MRb

### *32-Bit Floating-Point Absolute Value*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1110 0010 0000
```

**Description**

The absolute value of MRb is loaded into MRa. Only the sign bit of the operand is modified by the MABSF32 instruction.

```
if (MRb < 0) {MRa = -MRb};
        else {MRa =  MRb};
```

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | No | No |

The MSTF register flags are modified as follows:

```
NF = 0;
ZF = 0;
if ( MRa(30:23) == 0) ZF = 1;
```

**Pipeline**

This is a single-cycle instruction.

**Example**

```
MMOVIZ MR0, #-2.0 ; MR0 = -2.0 (0xC0000000)
MABSF32 MR0, MR0  ; MR0 = 2.0 (0x40000000), ZF = NF = 0
MMOVIZ MR0, #5.0  ; MR0 = 5.0 (0x40A00000)
MABSF32 MR0, MR0  ; MR0 = 5.0 (0x40A00000), ZF = NF = 0
MMOVIZ MR0, #0.0  ; MR0 = 0.0
MABSF32 MR0, MR0  ; MR0 = 0.0 ZF = 1, NF = 0
```

**See also**

MNEGF32 MRa, MRb {, CNDF}



Copyright © 2024 Texas Instruments Incorporated

### MADD32 MRa, MRb, MRc

***32-Bit Integer Add***

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point destination register (MR0 to MR3) |
| MRc | CLA floating-point destination register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 000cc bbaa
MSW: 0111 1110 1100 0000
```

**Description**

32-bit integer addition of MRb and MRc.

```
MRa(31:0) = MRb(31:0) + MRc(31:0);
```

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | No | No |

The MSTF register flags are modified based on the integer results of the operation.

```
NF = MRa(31);
ZF = 0;
if(MRa(31:0) == 0) { ZF = 1; };
```

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Given A = (int32)1
;       B = (int32)2
;       C = (int32)-7
;
; Calculate Y2 = A + B + C
;
_Cla1Task1:
       MMOV32 MR0, @_A      ; MR0 = 1 (0x00000001)
       MMOV32 MR1, @_B      ; MR1 = 2 (0x00000002)
       MMOV32 MR2, @_C      ; MR2 = -7 (0xFFFFFFF9)
       MADD32 MR3, MR0, MR1 ; A + B
       MADD32 MR3, MR2, MR3 ; A + B + C = -4 (0xFFFFFFFC)
       MMOV32 @_y2, MR3     ; Store y2
       MSTOP                ; end of task
```

**See also**

MAND32 MRa, MRb, MRc
MASR32 MRa, #SHIFT
MLSL32 MRa, #SHIFT
MLSR32 MRa, #SHIFT
MOR32 MRa, MRb, MRc
MXOR32 MRa, MRb, MRc
MSUB32 MRa, MRb, MRc



Copyright © 2024 Texas Instruments Incorporated

**MADDF32 MRa, #16FHi, MRb**

***32-Bit Floating-Point Addition***

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| #16FHi | A 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0. |
| MRb | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: IIII IIII IIII IIII
MSW: 0111 0111 1100 bbaa
```

**Description**

Add MRb to the floating-point value represented by the immediate operand. Store the result of the addition in MRa.

#16FHi is a 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0. #16FHi is most useful for representing constants where the lowest 16-bits of the mantissa are 0. Some examples are 2.0 (0x40000000), 4.0 (0x40800000), 0.5 (0x3F000000), and -1.5 (0xBFC00000). The assembler accepts either a hex or float as the immediate value. That is, the value -1.5 can be represented as #-1.5 or #0xBFC0.

```
MRa = MRb + #16FHi:0;
```

This instruction can also be written as MADDF32 MRa, MRb, #16FHi.

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | Yes | Yes |

The MSTF register flags are modified as follows:

- LUF = 1 if MADDF32 generates an underflow condition.
- LVF = 1 if MADDF32 generates an overflow condition.

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Add to MR1 the value 2.0 in 32-bit floating-point format
; Store the result in MR0
  MADDF32 MR0, #2.0, MR1    ; MR0 = 2.0 + MR1
; Add to MR3 the value -2.5 in 32-bit floating-point format
; Store the result in MR2
  MADDF32 MR2, #-2.5, MR3   ; MR2 = -2.5 + MR3
; Add to MR3 the value 0x3FC00000 (1.5)
; Store the result in MR3
  MADDF32 MR3, #0x3FC0, MR3 ; MR3 = 1.5 + MR3
```



Copyright © 2024 Texas Instruments Incorporated

**MADDF32 MRa, #16FHi, MRb** (continued)

***32-Bit Floating-Point Addition***

**See also**

MADDF32 MRa, MRb, #16FHi
MADDF32 MRa, MRb, MRc
MADDF32 MRd, MRe, MRf || MMOV32 MRa, mem32
MADDF32 MRd, MRe, MRf || MMOV32 mem32, MRa
MMPYF32 MRa, MRb, MRc || MADDF32 MRd, MRe, MRf



Copyright © 2024 Texas Instruments Incorporated

**MADDF32 MRa, MRb, #16FHi**

***32-Bit Floating-Point Addition***

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |
| #16FHi | A 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0. |

**Opcode**

```
LSW: IIII IIII IIII IIII
MSW: 0111 0111 1100 bbaa
```

**Description**

Add MRb to the floating-point value represented by the immediate operand. Store the result of the addition in MRa.

#16FHi is a 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0. #16FHi is most useful for representing constants where the lowest 16-bits of the mantissa are 0. Some examples are 2.0 (0x40000000), 4.0 (0x40800000), 0.5 (0x3F000000), and -1.5 (0xBFC00000). The assembler accepts either a hex or float as the immediate value. That is, the value -1.5 can be represented as #-1.5 or #0xBFC0.

```
MRa = MRb + #16FHi:0;
```

This instruction can also be written as MADDF32 MRa, #16FHi, MRb.

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | Yes | Yes |

The MSTF register flags are modified as follows:

- LUF = 1 if MADDF32 generates an underflow condition.
- LVF = 1 if MADDF32 generates an overflow condition.

**Pipeline**

This is a single-cycle instruction.



Copyright © 2024 Texas Instruments Incorporated

**MADDF32 MRa, MRb, #16FHi** (continued)

### *32-Bit Floating-Point Addition*

**Example 1**

```
; X is an array of 32-bit floating-point values
; Find the maximum value in an array X
; and store the value in Result
;
_Cla1Task1:
    MMOVI16    MAR1,#_X          ; Start address
    MUI16TOF32 MR0, @_len        ; Length of the array
    MNOP                         ; delay for MAR1 load
    MNOP                         ; delay for MAR1 load
    MMOV32     MR1, *MAR1[2]++   ; MR1 = X0
LOOP
    MMOV32     MR2, *MAR1[2]++   ; MR2 = next element
    MMAXF32    MR1,  MR2         ; MR1 = MAX(MR1, MR2)
    MADDF32    MR0, MR0, #-1.0   ; Decrement the counter
    MCMPF32    MR0 #0.0          ; Set/clear flags for MBCNDD
    MNOP
    MNOP
    MNOP
    MBCNDD LOOP, NEQ             ; Branch if not equal to zero
    MMOV32 @_Result, MR1         ; Always executed
    MNOP                         ; Always executed
    MNOP                         ; Always executed
    MSTOP                        ; End of task
```

**Example 2**

```
; Show the basic operation of MADDF32
;
; Add to MR1 the value 2.0 in 32-bit floating-point format
; Store the result in MR0
    MADDF32 MR0, MR1, #2.0    ; MR0 = MR1 + 2.0
; Add to MR3 the value -2.5 in 32-bit floating-point format
; Store the result in MR2
    MADDF32 MR2, MR3, #-2.5   ; MR2 = MR3 + (-2.5)
; Add to MR0 the value 0x3FC00000 (1.5)
; Store the result in MR0
   MADDF32 MR0, MR0, #0x3FC0  ; MR0 = MR0 + 1.5
```

**See also**

MADDF32 MRa, #16FHi, MRb
MADDF32 MRa, MRb, MRc
MADDF32 MRd, MRe, MRf || MMOV32 MRa, mem32
MADDF32 MRd, MRe, MRf || MMOV32 mem32, MRa
MMPYF32 MRa, MRb, MRc || MADDF32 MRd, MRe, MRf



Copyright © 2024 Texas Instruments Incorporated

## MADDF32 MRa, MRb, MRc

### *32-Bit Floating-Point Addition*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |
| MRc | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: 000 0000 00cc bbaa
MSW: 0111 1100 0010 0000
```

**Description**

Add the contents of MRc to the contents of MRb and load the result into MRa.

```
MRa = MRb + MRc;
```

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | Yes | Yes |

The MSTF register flags are modified as follows:

- LUF = 1 if MADDF32 generates an underflow condition.
- LVF = 1 if MADDF32 generates an overflow condition.

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Given M1, X1, and B1 are 32-bit floating-point numbers
; Calculate Y1 = M1*X1+B1
;
_Cla1Task1:
   MMOV32 MR0,@M1       ; Load MR0 with M1
   MMOV32 MR1,@X1       ; Load MR1 with X1
   MMPYF32 MR1,MR1,MR0  ; Multiply M1*X1
|| MMOV32 MR0,@B1       ; and in parallel load MR0 with B1
   MADDF32 MR1,MR1,MR0  ; Add M*X1 to B1 and store in MR1
   MMOV32 @Y1,MR1       ; Store the result
   MSTOP                ; end of task
```

**See also**

MADDF32 MRa, #16FHi, MRb
MADDF32 MRa, MRb, #16FHi
MADDF32 MRd, MRe, MRf || MMOV32 MRa, mem32
MADDF32 MRd, MRe, MRf || MMOV32 mem32, MRa
MMPYF32 MRa, MRb, MRc || MADDF32 MRd, MRe, MRf



Copyright © 2024 Texas Instruments Incorporated

**MADDF32 MRd, MRe, MRf||MMOV32 mem32, MRa**

***32-Bit Floating-Point Addition with Parallel Move***

## Operands

| | |
|---|---|
| MRd | CLA floating-point destination register for the MADDF32 (MR0 to MR3) |
| MRe | CLA floating-point source register for the MADDF32 (MR0 to MR3) |
| MRf | CLA floating-point source register for the MADDF32 (MR0 to MR3) |
| mem32 | 32-bit memory location accessed using one of the available addressing modes. This is the destination of the MMOV32. |
| MRa | CLA floating-point source register for the MMOV32 (MR0 to MR3) |

## Opcode

```
LSW: mmmm mmmm mmmm mmmm
MSW: 0101 ffee ddaa addr
```

## Description

Perform an MADDF32 and a MMOV32 in parallel. Add MRf to the contents of MRe and store the result in MRd. In parallel move the contents of MRa to the 32-bit location mem32.

```
MRd = MRe + MRf;
[mem32] = MRa;
```

## Flags

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | Yes | Yes |

The MSTF register flags are modified as follows:

- LUF = 1 if MADDF32 generates an underflow condition.
- LVF = 1 if MADDF32 generates an overflow condition.

## Pipeline

Both MADDF32 and MMOV32 complete in a single cycle.

## Example

```
; Given A, B, and C are 32-bit floating-point numbers
; Calculate Y2 = (A * B)
;           Y3 = (A * B) + C
;
_Cla1Task2:
   MMOV32   MR0, @_A         ; Load MR0 with A
   MMOV32   MR1, @_B         ; Load MR1 with B
   MMPYF32  MR1, MR1, MR0    ; Multiply A*B
|| MMOV32   MR0, @_C         ;     and in parallel load MR0 with C
   MADDF32  MR1, MR1, MR0    ; Add (A*B) to C
|| MMOV32   @_Y2, MR1        ;     and in parallel store A*B
   MMOV32   @_Y3, MR1        ; Store the A*B + C
   MSTOP                     ; end of task
```

## See also

MADDF32 MRa, #16FHi, MRb
MADDF32 MRa, MRb, #16FHi
MADDF32 MRa, MRb, MRc
MMPYF32 MRa, MRb, MRc || MADDF32 MRd, MRe, MRf
MADDF32 MRd, MRe, MRf || MMOV32 MRa, mem32



Copyright © 2024 Texas Instruments Incorporated

**MADDF32 MRd, MRe, MRf ||MMOV32 MRa, mem32**

***32-Bit Floating-Point Addition with Parallel Move***

## Operands

| | |
|---|---|
| MRd | CLA floating-point destination register for the MADDF32 (MR0 to MR3).<br>MRd cannot be the same register as MRa. |
| MRe | CLA floating-point source register for the MADDF32 (MR0 to MR3) |
| MRf | CLA floating-point source register for the MADDF32 (MR0 to MR3) |
| MRa | CLA floating-point destination register for the MMOV32 (MR0 to MR3).<br>MRa cannot be the same register as MRd. |
| mem32 | 32-bit memory location accessed using one of the available addressing modes. This is the source for the MMOV32. |

## Opcode

```
LSW: mmmm mmmm mmmm mmmm
MSW: 0001 ffee ddaa addr
```

## Description

Perform an MADDF32 and a MMOV32 operation in parallel. Add MRf to the contents of MRe and store the result in MRd. In parallel move the contents of the 32-bit location mem32 to MRa.

```
MRd = MRe + MRf;
MRa = [mem32];
```

## Restrictions

The destination register for the MADDF32 and the MMOV32 must be unique. That is, MRa and MRd cannot be the same register.

## Flags

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | Yes | Yes |

The MSTF register flags are modified as follows:

- LUF = 1 if MADDF32 generates an underflow condition.
- LVF = 1 if MADDF32 generates an overflow condition.

The MMOV32 Instruction sets the NF and ZF flags as follows:

```
NF = MRa(31);
ZF = 0;
if(MRa(30:23) == 0) { ZF = 1; NF = 0; };
```

## Pipeline

The MADDF32 and the MMOV32 both complete in a single cycle.



Copyright © 2024 Texas Instruments Incorporated

**MADDF32 MRd, MRe, MRf ||MMOV32 MRa, mem32** (continued)

### *32-Bit Floating-Point Addition with Parallel Move*

**Example 1**

```
; Given A, B, and C are 32-bit floating-point numbers
; Calculate Y1 = A + 4B
;           Y2 = A + C
;
_Cla1Task1:
    MMOV32 MR0, @A          ; Load MR0 with A
    MMOV32 MR1, @B          ; Load MR1 with B
    MMPYF32 MR1, MR1, #4.0 ; Multiply 4 * B
||  MMOV32 MR2, @C            and in parallel load C
    MADDF32 MR3, MR0, MR1  ; Add A + 4B
    MADDF32 MR3, MR0, MR2  ; Add A + C
||  MMOV32 @Y1, MR3         ; and in parallel store A+4B
    MMOV32 @Y2, MR3         ; store A + C MSTOP
                            ; end of task
```

**Example 2**

```
; Given A, B, and C are 32-bit floating-point numbers
; Calculate Y3 = (A + B)
;           Y4 = (A + B) * C
;
_Cla1Task2:
     MMOV32 MR0, @A          ; Load MR0 with A
     MMOV32 MR1, @B          ; Load MR1 with B
     MADDF32 MR1, MR1, MR0  ; Add A+B
||   MMOV32 MR0, @C          ; and in parallel load MR0 with C
     MMPYF32 MR1, MR1, MR0  ; Multiply (A+B) by C
||   MMOV32 @Y3, MR1         ; and in parallel store A+B
     MMOV32 @Y4, MR1         ; Store the (A+B) * C
     MSTOP                   ; end of task
```

**See also**

MADDF32 MRa, #16FHi, MRb
MADDF32 MRa, MRb, #16FHi
MADDF32 MRa, MRb, MRc
MADDF32 MRd, MRe, MRf || MMOV32 mem32, MRa
MMPYF32 MRa, MRb, MRc || MADDF32 MRd, MRe, MRf



Copyright © 2024 Texas Instruments Incorporated

## MAND32 MRa, MRb, MRc

### *Bitwise AND*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |
| MRc | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 00cc bbaa
MSW: 0111 1100 0110 0000
```

**Description**

Bitwise AND of MRb with MRc.

```
MRa(31:0) = MRb(31:0) AND MRc(31:0);
```

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | No | No |

The MSTF register flags are modified based on the integer results of the operation.

```
NF = MRa(31);
ZF = 0;
if(MRa(31:0) == 0) { ZF = 1; }
```

**Pipeline**

This is a single-cycle instruction.

**Example**

```
MMOVIZ   MR0,    #0x5555  ; MR0 = 0x5555AAAA
MMOVXI   MR0,    #0xAAAA
MMOVIZ   MR1,    #0x5432  ; MR1 = 0x5432FEDC
MMOVXI   MR1,    #0xFEDC
; 0101 AND 0101 = 0101 (5)
; 0101 AND 0100 = 0100 (4)
; 0101 AND 0011 = 0001 (1)
; 0101 AND 0010 = 0000 (0)
; 1010 AND 1111 = 1010 (A)
; 1010 AND 1110 = 1010 (A)
; 1010 AND 1101 = 1000 (8)
; 1010 AND 1100 = 1000 (8)
MAND32 MR2, MR1, MR0      ; MR3 = 0x5410AA88
```

**See also**

MADD32 MRa, MRb, MRc
MASR32 MRa, #SHIFT
MLSL32 MRa, #SHIFT
MLSR32 MRa, #SHIFT
MOR32 MRa, MRb, MRc
MXOR32 MRa, MRb, MRc
MSUB32 MRa, MRb, MRc



Copyright © 2024 Texas Instruments Incorporated

### MASR32 MRa, #SHIFT

***Arithmetic Shift Right***

**Operands**

| | |
|---|---|
| MRa | CLA floating-point source/destination register (MR0 to MR3) |
| #SHIFT | Number of bits to shift (1 to 32) |

**Opcode**

```
LSW: 0000 0000 0shi ftaa
MSW: 0111 1011 0100 0000
```

**Description**

Arithmetic shift right of MRa by the number of bits indicated. The number of bits can be 1 to 32.

```
MARa(31:0) = Arithmetic Shift(MARa(31:0) by #SHIFT bits);
```

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | No | No |

The MSTF register flags are modified based on the integer results of the operation.

```
NF = MRa(31);
ZF = 0;
if(MRa(31:0) == 0) { ZF = 1; }
```

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Given m2 = (int32)32
;       x2 = (int32)64
;       b2 = (int32)-128
;
; Calculate
;       m2 = m2/2
;       x2 = x2/4
;       b2 = b2/8
;
_Cla1Task2:
   MMOV32 MR0, @_m2 ; MR0 = 32 (0x00000020)
   MMOV32 MR1, @_x2 ; MR1 = 64 (0x00000040)
   MMOV32 MR2, @_b2 ; MR2 = -128 (0xFFFFFF80)
   MASR32 MR0, #1   ; MR0 = 16 (0x00000010)
   MASR32 MR1, #2   ; MR1 = 16 (0x00000010)
   MASR32 MR2, #3   ; MR2 = -16 (0xFFFFFFF0)
   MMOV32 @_m2, MR0 ; store results
   MMOV32 @_x2, MR1
   MMOV32 @_b2, MR2
   MSTOP ; end of task
```



Copyright © 2024 Texas Instruments Incorporated

**MASR32 MRa, #SHIFT** (continued)

***Arithmetic Shift Right***

**See also**

MADD32 MRa, MRb, MRc
MAND32 MRa, MRb, MRc
MLSL32 MRa, #SHIFT
MLSR32 MRa, #SHIFT
MOR32 MRa, MRb, MRc
MXOR32 MRa, MRb, MRc
MSUB32 MRa, MRb, MRc



Copyright © 2024 Texas Instruments Incorporated

**MBCNDD 16BitDest {, CNDF}**

***Branch Conditional Delayed***

## Operands

| | |
|---|---|
| 16BitDest | 16-bit destination if condition is true |
| CNDF | Optional condition tested |

## Opcode

```
LSW: dest dest dest dest
MSW: 0111 1001 1000 cndf
```

## Description

If the specified condition is true, then branch by adding the signed 16BitDest value to the MPC value. Otherwise, continue without branching. If the address overflows, the address wraps around. Therefore, a value of "0xFFFE" puts the MPC back to the MBCNDD instruction.

Refer to the Pipeline section for important information regarding this instruction.

```
if (CNDF == TRUE) MPC += 16BitDest;
```

CNDF is one of the following conditions:

| Encode(1) | CNDF | Description | MSTF Flags Tested |
|---|---|---|---|
| 0000 | NEQ | Not equal to zero | ZF == 0 |
| 0001 | EQ | Equal to zero | ZF == 1 |
| 0010 | GT | Greater than zero | ZF == 0 AND NF == 0 |
| 0011 | GEQ | Greater than or equal to zero | NF == 0 |
| 0100 | LT | Less than zero | NF == 1 |
| 0101 | LEQ | Less than or equal to zero | ZF == 1 OR NF == 1 |
| 1010 | TF | Test flag set | TF == 1 |
| 1011 | NTF | Test flag not set | TF == 0 |
| 1100 | LU | Latched underflow | LUF == 1 |
| 1101 | LV | Latched overflow | LVF == 1 |
| 1110 | UNC | Unconditional | None |
| 1111 | UNCF(2) | Unconditional with flag modification | None |

(1) Values not shown are reserved.
(2) This is the default operation, if no CNDF field is specified. This condition allows the ZF and NF flags to be modified when a conditional operation is executed. All other conditions do not modify these flags.

## Restrictions

The MBCNDD instruction is not allowed three instructions before or after a MBCNDD, MCCNDD, or MRCNDD instruction. Refer to the Pipeline section for more information.

## Flags

This instruction does not modify flags in the MSTF register.

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |



Copyright © 2024 Texas Instruments Incorporated

**MBCNDD 16BitDest {, CNDF}** (continued)

***Branch Conditional Delayed***

**Pipeline**

The MBCNDD instruction alone is a single-cycle instruction. As shown in Table 6-11, 6 instruction slots are executed for each branch; 3 slots before the branch instruction (I2-I4) and 3 slots after the branch instruction (I5-I7). The total number of cycles for a branch taken or not taken depends on the usage of these slots. That is, the number of cycles depends on how many slots are filled with a MNOP as well as which slots are filled. The effective number of cycles for a branch can, therefore, range from 1 to 7 cycles. The number of cycles for a branch taken cannot be the same as for a branch not taken.

Referring to Table 6-11 and Table 6-12, the instructions before and after MBCNDD have the following properties:

- **I1**
  - I1 is the last instruction that can effect the CNDF flags for the MBCNDD instruction. The CNDF flags are tested in the D2 phase of the pipeline. That is, a decision is made whether to branch or not when MBCNDD is in the D2 phase.
  - There are no restrictions on the type of instruction for I1.
- **I2, I3, and I4**
  - The three instructions proceeding MBCNDD can change MSTF flags but have no effect on whether the MBCNDD instruction branches or not. This is because the flag modification occurs after the D2 phase of the MBCNDD instruction.
  - These instructions must not be the following: MSTOP, MDEBUGSTOP, MBCNDD, MCCNDD, or MRCNDD.
- **I5, I6, and I7**
  - The three instructions following MBCNDD are always executed irrespective of whether the branch is taken or not.
  - These instructions must not be the following: MSTOP, MDEBUGSTOP, MBCNDD, MCCNDD, or MRCNDD.

```
<Instruction 1>   ; I1 Last instruction that can affect flags for
                  ; the MBCNDD operation
<Instruction 2>   ; I2 Cannot be stop, branch, call or return
<Instruction 3>   ; I3 Cannot be stop, branch, call or return
<Instruction 4>   ; I4 Cannot be stop, branch, call or return
MBCNDD _Skip, NEQ ; Branch to Skip if not eqal to zero
                  ; Three instructions after MBCNDD are always
                  ; executed whether the branch is taken or not
<Instruction 5>   ; I5 Cannot be stop, branch, call or return
<Instruction 6>   ; I6 Cannot be stop, branch, call or return
<Instruction 7>   ; I7 Cannot be stop, branch, call or return
<Instruction 8>   ; I8
<Instruction 9>   ; I9
....
_Skip:
 <Destination 1>  ; d1 Can be any instruction
 <Destination 2>  ; d2
 <Destination 3>  ; d3
....
....
MSTOP
....
```



Copyright © 2024 Texas Instruments Incorporated

**MBCNDD 16BitDest {, CNDF}** (continued)

### *Branch Conditional Delayed*

**Table 6-11. Pipeline Activity for MBCNDD, Branch Not Taken**

| Instruction | F1 | F2 | D1 | D2 | R1 | R2 | E | W |
|---|---|---|---|---|---|---|---|---|
| I1 | I1 | | | | | | | |
| I2 | I2 | I1 | | | | | | |
| I3 | I3 | I2 | I1 | | | | | |
| I4 | I4 | I3 | I2 | I1 | | | | |
| MBCNDD | MBCNDD | I4 | I3 | I2 | I1 | | | |
| I5 | I5 | MBCNDD | I4 | I3 | I2 | I1 | | |
| I6 | I6 | I5 | MBCNDD | I4 | I3 | I2 | I1 | |
| I7 | I7 | I6 | I5 | MBCNDD | I4 | I3 | I2 | |
| I8 | I8 | I7 | I6 | I5 | - | I4 | I3 | |
| I9 | I9 | I8 | I7 | I6 | I5 | - | I4 | |
| I10 | I10 | I9 | I8 | I7 | I6 | I5 | - | |
| | | I10 | I9 | I8 | I7 | I6 | I5 | |
| | | | I10 | I9 | I8 | I7 | I6 | |
| | | | | I10 | I9 | I8 | I7 | |
| | | | | | I10 | I9 | I8 | |
| | | | | | | I10 | I9 | |
| | | | | | | | I10 | |

**Table 6-12. Pipeline Activity for MBCNDD, Branch Taken**

| Instruction | F1 | F2 | D1 | D2 | R1 | R2 | E | W |
|---|---|---|---|---|---|---|---|---|
| I1 | I1 | | | | | | | |
| I2 | I2 | I1 | | | | | | |
| I3 | I3 | I2 | I1 | | | | | |
| I4 | I4 | I3 | I2 | I1 | | | | |
| MBCNDD | MBCNDD | I4 | I3 | I2 | I1 | | | |
| I5 | I5 | MBCNDD | I4 | I3 | I2 | I1 | | |
| I6 | I6 | I5 | MBCNDD | I4 | I3 | I2 | I1 | |
| I7 | I7 | I6 | I5 | MBCNDD | I4 | I3 | I2 | |
| d1 | d1 | I7 | I6 | I5 | - | I4 | I3 | |
| d2 | d2 | d1 | I7 | I6 | I5 | - | I4 | |
| d3 | d3 | d2 | d1 | I7 | I6 | I5 | - | |
| | | d3 | d2 | d1 | I7 | I6 | I5 | |
| | | | d3 | d2 | d1 | I7 | I6 | |
| | | | | d3 | d2 | d1 | I7 | |
| | | | | | d3 | d2 | d1 | |
| | | | | | | d3 | d2 | |
| | | | | | | | d3 | |



Copyright © 2024 Texas Instruments Incorporated

**MBCNDD 16BitDest {, CNDF}** (continued)

### *Branch Conditional Delayed*

**Example 1**

```
; if (State == 0.1)
; RampState = RampState || RAMPMASK
; else if (State == 0.01)
; CoastState = CoastState || COASTMASK
; else
; SteadyState = SteadyState || STEADYMASK
;
_Cla1Task1:
 MMOV32 MR0, @State
 MCMPF32 MR0, #0.1        ; Affects flags for 1st MBCNDD (A)
 MNOP
 MNOP
 MNOP
 MBCNDD Skip1, NEQ        ; (A) If State != 0.1, go to Skip1
 MNOP ; Always executed
 MNOP ; Always executed
 MNOP ; Always executed
 MMOV32 MR1, @RampState   ; Execute if (A) branch not taken
 MMOVXI MR2, #RAMPMASK    ; Execute if (A) branch not taken
 MOR32 MR1, MR2           ; Execute if (A) branch not taken
 MMOV32 @RampState, MR1   ; Execute if (A) branch not taken
 MSTOP                    ; end of task if (A) branch not taken
Skip1:
 MCMPF32 MR0,#0.01        ; Affects flags for 2nd MBCNDD (B)
 MNOP
 MNOP
 MNOP
 MBCNDD Skip2,NEQ         ; (B) If State != 0.01, go to Skip2
 MNOP ; Always executed
 MNOP ; Always executed
 MNOP ; Always executed
 MMOV32 MR1, @CoastState  ; Execute if (B) branch not taken
 MMOVXI MR2, #COASTMASK   ; Execute if (B) branch not taken
 MOR32 MR1, MR2           ; Execute if (B) branch not taken
 MMOV32 @CoastState, MR1  ; Execute if (B) branch not taken
 MSTOP
Skip2:
 MMOV32 MR3, @SteadyState ; Executed if (B) branch taken
 MMOVXI MR2, #STEADYMASK  ; Executed if (B) branch taken
 MOR32 MR3, MR2           ; Executed if (B) branch taken
 MMOV32 @SteadyState, MR3 ; Executed if (B) branch taken
 MSTOP
```



Copyright © 2024 Texas Instruments Incorporated

**MBCNDD 16BitDest {, CNDF}** (continued)

***Branch Conditional Delayed***

**Example 2**

```
; This example is the same as Example 1, except
; the code is optimized to take advantage of delay slots
;
; if (State == 0.1)
; RampState = RampState || RAMPMASK
; else if (State == 0.01)
; CoastState = CoastState || COASTMASK
; else
; SteadyState = SteadyState || STEADYMASK
;
_Cla1Task2:
 MMOV32 MR0, @State
 MCMPF32 MR0, #0.1          ; Affects flags for 1st MBCNDD (A)
 MCMPF32 MR0, #0.01         ; Check used by 2nd MBCNDD (B)
 MTESTTF EQ                 ; Store EQ flag in TF for 2nd MBCNDD (B)
 MNOP
 MBCNDD Skip1, NEQ          ; (A) If State != 0.1, go to Skip1
 MMOV32 MR1, @RampState     ; Always executed
 MMOVXI MR2, #RAMPMASK      ; Always executed
 MOR32 MR1, MR2             ; Always executed
 MMOV32 @RampState, MR1     ; Execute if (A) branch not taken
 MSTOP                      ; end of task if (A) branch not taken
Skip1:
 MMOV32 MR3, @SteadyState
 MMOVXI MR2, #STEADYMASK
 MOR32 MR3, MR2
 MBCNDD Skip2, NTF          ; (B) if State != .01, go to Skip2
 MMOV32 MR1, @CoastState    ; Always executed
 MMOVXI MR2, #COASTMASK     ; Always executed
 MOR32 MR1, MR2             ; Always executed
 MMOV32 @CoastState, MR1    ; Execute if (B) branch not taken
 MSTOP                      ; end of task if (B) branch not taken
Skip2:
 MMOV32 @SteadyState, MR3   ; Executed if (B) branch taken
 MSTOP
```

**See also**

MCCNDD 16BitDest, CNDF
MRCNDD CNDF



Copyright © 2024 Texas Instruments Incorporated

**MCCNDD 16BitDest {, CNDF}**

***Call Conditional Delayed***

**Operands**

| | |
|---|---|
| 16BitDest | 16-bit destination if condition is true |
| CNDF | Optional condition to be tested |

**Opcode**

```
LSW: dest dest dest dest
MSW: 0111 1001 1001 cndf
```

**Description**

If the specified condition is true, then store the return address in the RPC field of MSTF and make the call by adding the signed 16BitDest value to the MPC value. Otherwise, continue code execution without making the call. If the address overflows, the address wraps around. Therefore a value of "0xFFFE" puts the MPC back to the MCCNDD instruction.

Refer to the Pipeline section for important information regarding this instruction.

```
if (CNDF == TRUE)
{
    RPC = return address;
    MPC += 16BitDest;
};
```

CNDF is one of the following conditions:

| Encode[1] | CNDF | Description | MSTF Flags Tested |
|---|---|---|---|
| 0000 | NEQ | Not equal to zero | ZF == 0 |
| 0001 | EQ | Equal to zero | ZF == 1 |
| 0010 | GT | Greater than zero | ZF == 0 AND NF == 0 |
| 0011 | GEQ | Greater than or equal to zero | NF == 0 |
| 0100 | LT | Less than zero | NF == 1 |
| 0101 | LEQ | Less than or equal to zero | ZF == 1 OR NF == 1 |
| 1010 | TF | Test flag set | TF == 1 |
| 1011 | NTF | Test flag not set | TF == 0 |
| 1100 | LU | Latched underflow | LUF == 1 |
| 1101 | LV | Latched overflow | LVF == 1 |
| 1110 | UNC | Unconditional | None |
| 1111 | UNCF[2] | Unconditional with flag modification | None |

(1) Values not shown are reserved.
(2) This is the default operation if no CNDF field is specified. This condition allows the ZF and NF flags to be modified when a conditional operation is executed. All other conditions do not modify these flags.

**Restrictions**

The MCCNDD instruction is not allowed three instructions before or after a MBCNDD, MCCNDD, or MRCNDD instruction. Refer to the Pipeline section for more details.

**Flags**

This instruction does not modify flags in the MSTF register.



Copyright © 2024 Texas Instruments Incorporated

**MCCNDD 16BitDest {, CNDF}** (continued)

***Call Conditional Delayed***

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

The MCCNDD instruction alone is a single-cycle instruction. As shown in Table 6-13, 6 instruction slots are executed for each call; 3 before the call instruction (I2-I4) and 3 after the call instruction (I5-I7). The total number of cycles for a call taken or not taken depends on the usage of these slots. That is, the number of cycles depends on how many slots are filled with a MNOP as well as which slots are filled. The effective number of cycles for a call can, therefore, range from 1 to 7 cycles. The number of cycles for a call taken cannot be the same as for a call not taken.

Referring to the following code fragment and the pipeline diagrams in Table 6-13 and Table 6-14, the instructions before and after MCCNDD have the following properties:

- **I1**
  - I1 is the last instruction that can effect the CNDF flags for the MCCNDD instruction. The CNDF flags are tested in the D2 phase of the pipeline. That is, a decision is made whether to branch or not when MCCNDD is in the D2 phase.
  - There are no restrictions on the type of instruction for I1.
- **I2, I3, and I4**
  - The three instructions proceeding MCCNDD can change MSTF flags but have no effect on whether the MCCNDD instruction makes the call or not. This is because the flag modification occurs after the D2 phase of the MCCNDD instruction.
  - These instructions must not be the following: MSTOP, MDEBUGSTOP, MBCNDD, MCCNDD, or MRCNDD.
- **I5, I6, and I7**
  - The three instructions following MBCNDD are always executed irrespective of whether the branch is taken or not.
  - These instructions must not be the following: MSTOP, MDEBUGSTOP, MBCNDD, MCCNDD, or MRCNDD.



Copyright © 2024 Texas Instruments Incorporated

**MCCNDD 16BitDest {, CNDF}** (continued)

### *Call Conditional Delayed*

```
<Instruction 1>    ; I1 Last instruction that can affect flags for
                   ;    the MCCNDD operation
<Instruction 2>    ; I2 Cannot be stop, branch, call or return
<Instruction 3>    ; I3 Cannot be stop, branch, call or return
<Instruction 4>    ; I4 Cannot be stop, branch, call or return
MCCNDD _func, NEQ  ; Call to func if not eqal to zero
                   ; Three instructions after MCCNDD are always
                   ; executed whether the call is taken or not
<Instruction 5>    ; I5 Cannot be stop, branch, call or return
<Instruction 6>    ; I6 Cannot be stop, branch, call or return
<Instruction 7>    ; I7 Cannot be stop, branch, call or return
<Instruction 8>    ; I8 The address of this instruction is saved
                   ;    in the RPC field of the MSTF register.
                   ;    Upon return this value is loaded into MPC
                   ;    and fetching continues from this point.
<Instruction 9>    ; I9
....
_func:
<Destination 1>    ; d1 Can be any instruction
<Destination 2>    ; d2
<Destination 3>    ; d3
<Destination 4>    ; d4 Last instruction that can affect flags for
                   ;    the MRCNDD operation
<Destination 5>    ; d5 Cannot be stop, branch, call or return
<Destination 6>    ; d6 Cannot be stop, branch, call or return
<Destination 7>    ; d7 Cannot be stop, branch, call or return
MRCNDD UNC         ; Return to <Instruction 8>, unconditional
                   ; Three instructions after MRCNDD are always
                   ; executed whether the return is taken or not
<Destination 8>    ; d8 Cannot be stop, branch, call or return
<Destination 9>    ; d9 Cannot be stop, branch, call or return
<Destination 10>   ; d10 Cannot be stop, branch, call or return
<Destination 11>   ; d11
....
MSTOP
```

**Table 6-13. Pipeline Activity for MCCNDD, Call Not Taken**

| Instruction | F1 | F2 | D1 | D2 | R1 | R2 | E | W |
|---|---|---|---|---|---|---|---|---|
| I1 | I1 | | | | | | | |
| I2 | I2 | I1 | | | | | | |
| I3 | I3 | I2 | I1 | | | | | |
| I4 | I4 | I3 | I2 | I1 | | | | |
| MCCNDD | MCCNDD | I4 | I3 | I2 | I1 | | | |
| I5 | I5 | MCCNDD | I4 | I3 | I2 | I1 | | |
| I6 | I6 | I5 | MCCNDD | I4 | I3 | I2 | I1 | |
| I7 | I7 | I6 | I5 | MCCNDD | I4 | I3 | I2 | |
| I8 | I8 | I7 | I6 | I5 | - | I4 | I3 | |
| I9 | I9 | I8 | I7 | I6 | I5 | - | I4 | |
| I10 | I10 | I9 | I8 | I7 | I6 | I5 | - | |
| etc .... | | I10 | I9 | I8 | I7 | I6 | I5 | |
| .... | | | I10 | I9 | I8 | I7 | I6 | |
| .... | | | | I10 | I9 | I8 | I7 | |
| .... | | | | | I10 | I9 | I8 | |
| | | | | | | I10 | I9 | |
| | | | | | | | I10 | |



Copyright © 2024 Texas Instruments Incorporated

**MCCNDD 16BitDest {, CNDF}** (continued)

### *Call Conditional Delayed*

**Table 6-14. Pipeline Activity for MCCNDD, Call Taken**

| Instruction | F1 | F2 | D1 | D2 | R1 | R2 | E | W |
|---|---|---|---|---|---|---|---|---|
| I1 | I1 | | | | | | | |
| I2 | I2 | I1 | | | | | | |
| I3 | I3 | I2 | I1 | | | | | |
| I4 | I4 | I3 | I2 | I1 | | | | |
| MCCNDD | MCCNDD | I4 | I3 | I2 | I1 | | | |
| I5 | I5 | MCCNDD | I4 | I3 | I2 | I1 | | |
| I6 | I6 | I5 | MCCNDD | I4 | I3 | I2 | I1 | |
| I7 (1) | I7 | I6 | I5 | MCCNDD | I4 | I3 | I2 | |
| d1 | d1 | I7 | I6 | I5 | - | I4 | I3 | |
| d2 | d2 | d1 | I7 | I6 | I5 | - | I4 | |
| d3 | d3 | d2 | d1 | I7 | I6 | I5 | - | |
| etc .... | | d3 | d2 | d1 | I7 | I6 | I5 | |
| .... | | | d3 | d2 | d1 | I7 | I6 | |
| .... | | | | d3 | d2 | d1 | I7 | |
| .... | | | | | d3 | d2 | d1 | |
| | | | | | | d3 | d2 | |
| | | | | | | | d3 | |

(1) The RPC value in the MSTF register points to the instruction following I7 (instruction I8).

**See also**

MBCNDD #16BitDest, CNDF
MMOV32 mem32, MSTF
MMOV32 MSTF, mem32
MRCNDD CNDF



Copyright © 2024 Texas Instruments Incorporated

**MCMP32 MRa, MRb**

***32-Bit Integer Compare for Equal, Less Than or Greater Than***

**Operands**

| | |
|---|---|
| MRa | CLA floating-point source register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1111 0010 0000
```

**Description**

Set ZF and NF flags on the result of MRa - MRb where MRa and MRb are 32-bit integers. For a floating-point compare, refer to MCMPF32.

**Note**

A known hardware issue exists in the MCMP32 instruction. Signed-integer comparisons using MCMP32 alone set the status bits in a way that is not useful for comparison when the difference between the two operands is too large, such as when the inputs have opposite sign and are near the extreme 32-bit signed values. This affects both signed and unsigned integer comparisons.

The compiler (version 18.1.5.LTS or higher) has implemented a workaround for this issue. The compiler checks the upper bits of the operands by performing a floating point comparison before proceeding to do the integer comparison or subtraction.

The compiler flag --cla_signed_compare_workaround enables this workaround.

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | No | No |

The MSTF register flags are modified based on the integer results of the operation.

```
If(MRa ==    MRb) {ZF=1; NF=0;}
If(MRa >  MRb) {ZF=0; NF=0;}
If(MRa <  MRb) {ZF=0; NF=1;}
```

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Behavior of ZF and NF flags for different comparisons
;
; Given A = (int32)1
;       B = (int32)2
;       C = (int32)-7
;
    MMOV32 MR0, @_A ; MR0 = 1 (0x00000001)
    MMOV32 MR1, @_B ; MR1 = 2 (0x00000002)
    MMOV32 MR2, @_C ; MR2 = -7 (0xFFFFFFF9)
    MCMP32 MR2, MR2 ; NF = 0, ZF = 1
    MCMP32 MR0, MR1 ; NF = 1, ZF = 0
    MCMP32 MR1, MR0 ; NF = 0, ZF = 0
```



Copyright © 2024 Texas Instruments Incorporated

**MCMP32 MRa, MRb** (continued)

***32-Bit Integer Compare for Equal, Less Than or Greater Than***

**See also**

MADD32 MRa, MRb, MRc
MSUB32 MRa, MRb, MRc



Copyright © 2024 Texas Instruments Incorporated

**MCMPF32 MRa, MRb**

***32-Bit Floating-Point Compare for Equal, Less Than or Greater Than***

**Operands**

| | |
|---|---|
| MRa | CLA floating-point source register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1101 0000 0000
```

**Description**

Set ZF and NF flags on the result of MRa - MRb. The MCMPF32 instruction is performed as a logical compare operation. This is possible because of the IEEE format offsetting the exponent. Basically the bigger the binary number, the bigger the floating-point value.

Special cases for inputs:

- Negative zero is treated as positive zero.
- A denormalized value is treated as positive zero.
- Not-a-Number (NaN) is treated as infinity.

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | No | No |

The MSTF register flags are modified as follows:

```
If(MRa == MRb)   {ZF=1; NF=0;}
If(MRa > MRb) {ZF=0; NF=0;}
If(MRa < MRb) {ZF=0; NF=1;}
```

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Behavior of ZF and NF flags for different comparisons
    MMOVIZ    MR1, #-2.0  ; MR1 = -2.0 (0xC0000000)
    MMOVIZ    MR0, #5.0   ; MR0 = 5.0 (0x40A00000)
    MCMPF32   MR1, MR0    ; ZF = 0, NF = 1
    MCMPF32   MR0, MR1    ; ZF = 0, NF = 0
    MCMPF32   MR0, MR0    ; ZF = 1, NF = 0
```

**See also**

MCMPF32 MRa, #16FHi
MMAXF32 MRa, #16FHi
MMAXF32 MRa, MRb
MMINF32 MRa, #16FHi
MMINF32 MRa, MRb



Copyright © 2024 Texas Instruments Incorporated

**MCMPF32 MRa, #16FHi**

## *32-Bit Floating-Point Compare for Equal, Less Than or Greater Than*

### Operands

| | |
|---|---|
| MRa | CLA floating-point source register (MR0 to MR3) |
| #16FHi | A 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0. |

### Opcode

```
LSW: IIII IIII IIII IIII
MSW: 0111 1000 1100 00aa
```

### Description

Compare the value in MRa with the floating-point value represented by the immediate operand. Set the ZF and NF flags on (MRa - #16FHi:0).

#16FHi is a 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0. This addressing mode is most useful for constants where the lowest 16-bits of the mantissa are 0. Some examples are 2.0 (0x40000000), 4.0 (0x40800000), 0.5 (0x3F000000), and -1.5 (0xBFC00000). The assembler accepts either a hex or float as the immediate value. That is, -1.5 can be represented as #-1.5 or #0xBFC0.

The MCMPF32 instruction is performed as a logical compare operation. This is possible because of the IEEE floating-point format offsets the exponent. Basically the bigger the binary number, the bigger the floating-point value.

Special cases for inputs:

- Negative zero is treated as positive zero.
- Denormalized value is treated as positive zero.
- Not-a-Number (NaN) is treated as infinity.

### Flags

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | No | No |

The MSTF register flags are modified as follows:

```
If(MRa == #16FHi:0) {ZF=1, NF=0;}
If(MRa > #16FHi:0) {ZF=0, NF=0;}
If(MRa < #16FHi:0) {ZF=0, NF=1;}
```

### Pipeline

This is a single-cycle instruction

### Example 1

```
; Behavior of ZF and NF flags for different comparisons
  MMOVIZ      MR1, #-2.0  ; MR1 = -2.0 (0xC0000000)
  MMOVIZ      MR0, #5.0   ; MR0 = 5.0 (0x40A00000)
  MCMPF32     MR1, #-2.2  ; ZF = 0, NF = 0
  MCMPF32     MR0, #6.5   ; ZF = 0, NF = 1
  MCMPF32     MR0, #5.0   ; ZF = 1, NF = 0
```



Copyright © 2024 Texas Instruments Incorporated

**MCMPF32 MRa, #16FHi** (continued)

### *32-Bit Floating-Point Compare for Equal, Less Than or Greater Than*

**Example 2**

```
; X is an array of 32-bit floating-point values
; and has length elements. Find the maximum value in
; the array and store the value in Result
;
; Note: MCMPF32 and MSWAPF can be replaced with MMAXF32
;
_Cla1Task1:
  MMOVI16 MAR1,#_X         ; Start address
  MUI16TOF32 MR0, @_len    ; Length of the array
  MNOP                     ; delay for MAR1 load
  MNOP                     ; delay for MAR1 load
  MMOV32 MR1, *MAR1[2]++   ; MR1 = X0
LOOP
  MMOV32 MR2, *MAR1[2]++   ; MR2 = next element
  MCMPF32 MR2, MR1         ; Compare MR2 with MR1
  MSWAPF MR1, MR2, GT      ; MR1 = MAX(MR1, MR2)
  MADDF32 MR0, MR0, #-1.0 ; Decrememt the counter
  MCMPF32 MR0 #0.0         ; Set/clear flags for MBCNDD
  MNOP
  MNOP
  MNOP
  MBCNDD LOOP, NEQ         ; Branch if not equal to zero
  MMOV32 @_Result, MR1     ; Always executed
  MNOP                     ; Always executed
  MNOP                     ; Always executed
  MSTOP                    ; End of task
```

**See also**

MCMPF32 MRa, MRb
MMAXF32 MRa, #16FHi
MMAXF32 MRa, MRb
MMINF32 MRa, #16FHi
MMINF32 MRa, MRb



Copyright © 2024 Texas Instruments Incorporated

**MDEBUGSTOP**

***Debug Stop Task***

**Operands**

| | |
|---|---|
| none | This instruction does not have any operands |

**Opcode**

```
LSW: 0000 0000 0000 0000
MSW: 0111 1111 0110 0000
```

**Description**

When CLA breakpoints are enabled, the MDEBUGSTOP instruction is used to halt a task so that the task can be debugged. That is, MDEBUGSTOP is the CLA breakpoint. If CLA breakpoints are not enabled, the MDEBUGSTOP instruction behaves like a MNOP. Unlike the MSTOP, the MIRUN flag is not cleared and an interrupt is not issued. A single-step or run operation continues execution of the task.

**Restrictions**

The MDEBUGSTOP instruction cannot be placed 3 instructions before or after a MBCNDD, MCCNDD, or MRCNDD instruction.

**Flags**

This instruction does not modify flags in the MSTF register.

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.

**See also**

MSTOP



Copyright © 2024 Texas Instruments Incorporated

## MEALLOW

***Enable CLA Write Access to EALLOW Protected Registers***

### Operands

| | |
|---|---|
| none | This instruction does not have any operands |

### Opcode

```
LSW: 0000 0000 0000 0000
MSW: 0111 1111 1001 0000
```

### Description

This instruction sets the MEALLOW bit in the CLA status register MSTF. When this bit is set, the CLA is allowed write access to EALLOW protected registers. To again protect against CLA writes to protected registers, use the MEDIS instruction.

MEALLOW and MEDIS only control CLA write access; reads are allowed even if MEALLOW has not been executed. MEALLOW and MEDIS are also independent from the main CPU's EALLOW/EDIS. This instruction does not modify the EALLOW bit in the main CPU's status register. The MEALLOW bit in MSTF only controls access for the CLA while the EALLOW bit in the ST1 register only controls access for the main CPU.

As with EALLOW, the MEALLOW bit is overridden using the JTAG port, allowing full control of register accesses during debug from Code Composer Studio.

### Flags

This instruction does not modify flags in the MSTF register.

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

### Pipeline

This is a single-cycle instruction.

### Example

```
; C header file including definition of
; the EPwm1Regs structure
;
; The ePWM TZSEL register is EALLOW protected
;
   .cdecls C,LIST,"CLAShared.h"
   ...
_Cla1Task1:
   ...
   MEALLOW                          ; Allow CLA write access
   MMOV16 @_EPwm1Regs.TZSEL.all, MR3 ; Write to TZSEL
   MEDIS                            ; Disallow CLA write access
   ...
   ...
   MSTOP
```

### See also

MEDIS



Copyright © 2024 Texas Instruments Incorporated

## MEDIS

### *Disable CLA Write Access to EALLOW Protected Registers*

**Operands**

| | |
|---|---|
| none | This instruction does not have any operands |

**Opcode**

```
LSW: 0000 0000 0000 0000
MSW: 0111 1111 1011 0000
```

**Description**

This instruction clears the MEALLOW bit in the CLA status register MSTF. When this bit is clear, the CLA is not allowed write access to EALLOW-protected registers. To enable CLA writes to protected registers, use the MEALLOW instruction.

MEALLOW and MEDIS only control CLA write access; reads are allowed even if MEALLOW has not been executed. MEALLOW and MEDIS are also independent from the main CPU's EALLOW/EDIS. This instruction does not modify the EALLOW bit in the main CPU's status register. The MEALLOW bit in MSTF only controls access for the CLA while the EALLOW bit in the ST1 register only controls access for the main CPU.

As with EALLOW, the MEALLOW bit is overridden using the JTAG port, allowing full control of register accesses during debug from the Code Composer Studio™ IDE.

**Flags**

This instruction does not modify flags in the MSTF register.

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; C header file including definition of
; the EPwm1Regs structure
;
; The ePWM TZSEL register is EALLOW protected
;
   .cdecls C,LIST,"CLAShared.h"
   ...
_Cla1Task1:
   ...
   MEALLOW                          ; Allow CLA write access
   MMOV16 @_EPwm1Regs.TZSEL.all, MR3 ; Write to TZSEL
   MEDIS                            ; Disallow CLA write access
   ...
   ...
   MSTOP
```

**See also**

MEALLOW



Copyright © 2024 Texas Instruments Incorporated

**MEINVF32 MRa, MRb**

***32-Bit Floating-Point Reciprocal Approximation***

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1111 0000 0000
```

**Description**

This operation generates an estimate of 1/X in 32-bit floating-point format accurate to approximately 8 bits. This value can be used in a Newton-Raphson algorithm to get a more accurate answer. That is:

```
Ye = Estimate(1/X);
Ye = Ye*(2.0 - Ye*X);
Ye = Ye*(2.0 - Ye*X);
```

After two iterations of the Newton-Raphson algorithm, you get an exact answer accurate to the 32-bit floating-point format. On each iteration, the mantissa bit accuracy approximately doubles. The MEINVF32 operation does not generate a negative zero, DeNorm, or NaN value.

```
MRa = Estimate of 1/MRb;
```

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | Yes | Yes |

The MSTF register flags are modified as follows:

- LUF = 1 if MEINVF32 generates an underflow condition.
- LVF = 1 if MEINVF32 generates an overflow condition.

**Pipeline**

This is a single-cycle instruction.



Copyright © 2024 Texas Instruments Incorporated

**MEINVF32 MRa, MRb** (continued)

### *32-Bit Floating-Point Reciprocal Approximation*

**Example**

```
; Calculate Num/Den using a Newton-Raphson algorithum for 1/Den
; Ye = Estimate(1/X)
; Ye = Ye*(2.0 - Ye*X)
; Ye = Ye*(2.0 - Ye*X)
;
_Cla1Task1:
    MMOV32 MR1, @_Den       ; MR1 = Den
    MEINVF32 MR2, MR1       ; MR2 = Ye = Estimate(1/Den)
    MMPYF32 MR3, MR2, MR1   ; MR3 = Ye*Den
    MSUBF32 MR3, #2.0, MR3  ; MR3 = 2.0 - Ye*Den
    MMPYF32 MR2, MR2, MR3   ; MR2 = Ye = Ye*(2.0 - Ye*Den)
    MMPYF32 MR3, MR2, MR1   ; MR3 = Ye*Den
 || MMOV32 MR0, @_Num       ; MR0 = Num
    MSUBF32 MR3, #2.0, MR3  ; MR3 = 2.0 - Ye*Den
    MMPYF32 MR2, MR2, MR3   ; MR2 = Ye = Ye*(2.0 - Ye*Den)
 || MMOV32 MR1, @_Den       ; Reload Den To Set Sign
    MNEGF32 MR0, MR0, EQ    ; if(Den == 0.0) Change Sign Of Num
    MMPYF32 MR0, MR2, MR0   ; MR0 = Y = Ye*Num
    MMOV32 @_Dest, MR0      ; Store result
    MSTOP                   ; end of task
```

**See also**

MEISQRTF32 MRa, MRb



Copyright © 2024 Texas Instruments Incorporated

**MEISQRTF32 MRa, MRb**

***32-Bit Floating-Point Square-Root Reciprocal Approximation***

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1110 0100 0000
```

**Description**

This operation generates an estimate of 1/sqrt(X) in 32-bit floating-point format accurate to approximately 8 bits. This value can be used in a Newton-Raphson algorithm to get a more accurate answer. That is:

```
Ye = Estimate(1/sqrt(X));
Ye = Ye*(1.5 - Ye*Ye*X/2.0);
Ye = Ye*(1.5 - Ye*Ye*X/2.0);
```

After 2 iterations of the Newton-Raphson algorithm, you get an exact answer accurate to the 32-bit floating-point format. On each iteration, the mantissa bit accuracy approximately doubles. The MEISQRTF32 operation does not generate a negative zero, DeNorm, or NaN value.

```
MRa = Estimate of 1/sqrt (MRb);
```

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | Yes | Yes |

The MSTF register flags are modified as follows:

- LUF = 1 if MEISQRTF32 generates an underflow condition.
- LVF = 1 if MEISQRTF32 generates an overflow condition.

**Pipeline**

This is a single-cycle instruction.



Copyright © 2024 Texas Instruments Incorporated

**MEISQRTF32 MRa, MRb** (continued)

### *32-Bit Floating-Point Square-Root Reciprocal Approximation*

**Example**

```
; Y = sqrt(X)
; Ye = Estimate(1/sqrt(X));
; Ye = Ye*(1.5 - Ye*Ye*X*0.5)
; Ye = Ye*(1.5 - Ye*Ye*X*0.5)
; Y = X*Ye
;
_Cla1Task3:
    MMOV32 MR0, @_x          ; MR0 = X
    MEISQRTF32 MR1, MR0      ; MR1 = Ye = Estimate(1/sqrt(X))
    MMOV32 MR1, @_x, EQ      ; if(X == 0.0) Ye = 0.0
    MMPYF32 MR3, MR0, #0.5   ; MR3 = X*0.5
    MMPYF32 MR2, MR1, MR3    ; MR2 = Ye*X*0.5
    MMPYF32 MR2, MR1, MR2    ; MR2 = Ye*Ye*X*0.5
    MSUBF32 MR2, #1.5, MR2   ; MR2 = 1.5 - Ye*Ye*X*0.5
    MMPYF32 MR1, MR1, MR2    ; MR1 = Ye = Ye*(1.5 - Ye*Ye*X*0.5)
    MMPYF32 MR2, MR1, MR3    ; MR2 = Ye*X*0.5
    MMPYF32 MR2, MR1, MR2    ; MR2 = Ye*Ye*X*0.5
    MSUBF32 MR2, #1.5, MR2   ; MR2 = 1.5 - Ye*Ye*X*0.5
    MMPYF32 MR1, MR1, MR2    ; MR1 = Ye = Ye*(1.5 - Ye*Ye*X*0.5)
    MMPYF32 MR0, MR1, MR0    ; MR0 = Y = Ye*X
    MMOV32 @_y, MR0          ; Store Y = sqrt(X)
    MSTOP                    ; end of task
```

**See also**

MEINVF32 MRa, MRb



Copyright © 2024 Texas Instruments Incorporated

**MF32TOI16 MRa, MRb**

***Convert 32-Bit Floating-Point Value to 16-Bit Integer***

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1101 1110 0000
```

**Description**

Convert a 32-bit floating point value in MRb to a 16-bit integer and truncate. The result is stored in MRa.

```
MRa(15:0) = F32TOI16(MRb);
MRa(31:16) = sign extension of MRa(15);
```

**Flags**

This instruction does not affect any flags:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.

**Example**

```
MMOVIZ      MR0, #5.0   ; MR0        = 5.0 (0x40A00000)
MF32TOI16   MR1, MR0    ; MR1(15:0)  = MF32TOI16(MR0) = 0x0005
                        ; MR1(31:16) = Sign extension of MR1(15) = 0x0000
MMOVIZ      MR2, #-5.0  ; MR2        = -5.0 (0xC0A00000)
MF32TOI16   MR3, MR2    ; MR3(15:0)  = MF32TOI16(MR2) = -5 (0xFFFB)
                        ; MR3(31:16) = Sign extension of MR3(15) = 0xFFFF
```

**See also**

MF32TOI16R MRa, MRb
MF32TOUI16 MRa, MRb
MF32TOUI16R MRa, MRb
MI16TOF32 MRa, MRb
MI16TOF32 MRa, mem16
MUI16TOF32 MRa, mem16
MUI16TOF32 MRa, MRb



Copyright © 2024 Texas Instruments Incorporated

### MF32TOI16R MRa, MRb

***Convert 32-Bit Floating-Point Value to 16-Bit Integer and Round***

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1110 0110 0000
```

**Description**

Convert the 32-bit floating point value in MRb to a 16-bit integer and round to the nearest even value. The result is stored in MRa.

```
MRa(15:0) = F32TOI16round(MRb);
MRa(31:16) = sign extension of MRa(15);
```

**Flags**

This instruction does not affect any flags:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.

**Example**

```
MMOVIZ MR0, #0x3FD9    ; MR0(31:16) = 0x3FD9
MMOVXI MR0, #0x999A    ; MR0(15:0) = 0x999A
                       ; MR0 = 1.7 (0x3FD9999A)
MF32TOI16R MR1, MR0    ; MR1(15:0) = MF32TOI16round (MR0) = 2 (0x0002)
                       ; MR1(31:16) = Sign extension of MR1(15) = 0x0000
MMOVF32 MR2, #-1.7     ; MR2 = -1.7 (0xBFD9999A)
MF32TOI16R MR3, MR2    ; MR3(15:0) = MF32TOI16round (MR2) = -2 (0xFFFE)
                       ; MR3(31:16) = Sign extension of MR2(15) = 0xFFFF
```

**See also**

MF32TOI16 MRa, MRb
MF32TOUI16 MRa, MRb
MF32TOUI16R MRa, MRb
MI16TOF32 MRa, MRb
MI16TOF32 MRa, mem16
MUI16TOF32 MRa, mem16
MUI16TOF32 MRa, MRb



Copyright © 2024 Texas Instruments Incorporated

**MF32TOI32 MRa, MRb**

***Convert 32-Bit Floating-Point Value to 32-Bit Integer***

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1101 0110 0000
```

**Description**

Convert the 32-bit floating-point value in MRb to a 32-bit integer value and truncate. Store the result in MRa.

```
MRa = F32TOI32(MRb);
```

**Flags**

This instruction does not affect any flags:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.

**Example 1**

```
MMOVF32    MR2, #11204005.0   ; MR2 = 11204005.0 (0x4B2AF5A5)
MF32TOI32  MR3, MR2           ; MR3 = MF32TOI32(MR2) = 11204005 (0x00AAF5A5)
MMOVF32    MR0, #-11204005.0  ; MR0 = -11204005.0 (0xCB2AF5A5)
MF32TOI32  MR1, MR0           ; MR1 = MF32TOI32(MR0) = -11204005 (0xFF550A5B)
```

**Example 2**

```
; Given X, M and B are IQ24 numbers:
; X = IQ24(+2.5) = 0x02800000
; M = IQ24(+1.5) = 0x01800000
; B = IQ24(-0.5) = 0xFF800000
;
; Calculate Y = X * M + B
;
; Convert M, X, and B from IQ24 to float
;
_Cla1Task2:
   MI32TOF32 MR0, @_M          ; MR0 = 0x4BC00000
   MI32TOF32 MR1, @_X          ; MR1 = 0x4C200000
   MI32TOF32 MR2, @_B          ; MR2 = 0xCB000000
   MMPYF32   MR0, MR0, #0x3380 ; M = 1/(1*2^24) * iqm = 1.5 (0x3FC00000)
   MMPYF32   MR1, MR1, #0x3380 ; X = 1/(1*2^24) * iqx = 2.5 (0x40200000)
   MMPYF32   MR2, MR2, #0x3380 ; B = 1/(1*2^24) * iqb = -.5 (0xBF000000)
   MMPYF32   MR3, MR0, MR1     ; M*X
   MADDF32   MR2, MR2, MR3     ; Y=MX+B = 3.25 (0x40500000)
; Convert Y from float32 to IQ24
   MMPYF32 MR2, MR2, #0x4B80   ; Y * 1*2^24
   MF32TOI32 MR2, MR2          ; IQ24(Y) = 0x03400000
   MMOV32 @_Y, MR2             ; store result
   MSTOP                       ; end of task
```



Copyright © 2024 Texas Instruments Incorporated

**MF32TOI32 MRa, MRb** (continued)

***Convert 32-Bit Floating-Point Value to 32-Bit Integer***

**See also**

MF32TOUI32 MRa, MRb
MI32TOF32 MRa, MRb
MI32TOF32 MRa, mem32
MUI32TOF32 MRa, MRb
MUI32TOF32 MRa, mem32



Copyright © 2024 Texas Instruments Incorporated

### MF32TOUI16 MRa, MRb

***Convert 32-Bit Floating-Point Value to 16-bit Unsigned Integer***

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1110 1010 0000
```

**Description**

Convert the 32-bit floating point value in MRb to an unsigned 16-bit integer value and truncate to zero. The result is stored in MRa. To instead round the integer to the nearest even value, use the MF32TOUI16R instruction.

```
MRa(15:0) = F32TOUI16(MRb);
MRa(31:16) = 0x0000;
```

**Flags**

This instruction does not affect any flags:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.

**Example**

```
MMOVIZ      MR0, #9.0     ; MR0 = 9.0 (0x41100000)
MF32TOUI16  MR1, MR0      ; MR1(15:0) = MF32TOUI16(MR0) = 9 (0x0009)
                          ; MR1(31:16) = 0x0000
MMOVIZ      MR2, #-9.0    ; MR2 = -9.0 (0xC1100000)
MF32TOUI16  MR3, MR2      ; MR3(15:0) = MF32TOUI16(MR2) = 0 (0x0000)
                          ; MR3(31:16) = 0x0000
```

**See also**

MF32TOI16 MRa, MRb
MF32TOUI16 MRa, MRb
MF32TOUI16R MRa, MRb
MI16TOF32 MRa, MRb
MI16TOF32 MRa, mem16
MUI16TOF32 MRa, mem16
MUI16TOF32 MRa, MRb



Copyright © 2024 Texas Instruments Incorporated

### MF32TOUI16R MRa, MRb

***Convert 32-Bit Floating-Point Value to 16-bit Unsigned Integer and Round***

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1110 1100 0000
```

**Description**

Convert the 32-bit floating-point value in MRb to an unsigned 16-bit integer and round to the closest even value. The result is stored in MRa. To instead truncate the converted value, use the MF32TOUI16 instruction.

```
MRa(15:0)  = MF32TOUI16round(MRb);
MRa(31:16) = 0x0000;
```

**Flags**

This instruction does not affect any flags:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.

**Example**

```
MMOVIZ       MR0, #0x412C     ; MR0 = 0x412C
MMOVXI       MR0, #0xCCCD     ; MR0 = 0xCCCD ; MR0 = 10.8 (0x412CCCCD)
MF32TOUI16R  MR1, MR0         ; MR1(15:0) = MF32TOUI16round(MR0) = 11 (0x000B)
                              ; MR1(31:16) = 0x0000
MMOVF32      MR2, #-10.8      ; MR2 = -10.8 (0x0xC12CCCCD)
MF32TOUI16R  MR3, MR2         ; MR3(15:0) = MF32TOUI16round(MR2) = 0 (0x0000)
                              ; MR3(31:16) = 0x0000
```

**See also**

MF32TOI16 MRa, MRb
MF32TOI16R MRa, MRb
MF32TOUI16 MRa, MRb
MI16TOF32 MRa, MRb
MI16TOF32 MRa, mem16
MUI16TOF32 MRa, mem16
MUI16TOF32 MRa, MRb



Copyright © 2024 Texas Instruments Incorporated

### MF32TOUI32 MRa, MRb

***Convert 32-Bit Floating-Point Value to 32-Bit Unsigned Integer***

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1101 1010 0000
```

**Description**

Convert the 32-bit floating-point value in MRb to an unsigned 32-bit integer and store the result in MRa.

```
MRa = F32TOUI32(MRb);
```

**Flags**

This instruction does not affect any flags:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.

**Example**

```
MMOVIZ      MR0, #12.5    ; MR0 = 12.5 (0x41480000)
MF32TOUI32  MR0, MR0      ; MR0 = MF32TOUI32 (MR0) = 12 (0x0000000C)
MMOVIZ      MR1, #-6.5    ; MR1 = -6.5 (0xC0D00000)
MF32TOUI32  MR2, MR1      ; MR2 = MF32TOUI32 (MR1) = 0.0 (0x00000000)
```

**See also**

MF32TOI32 MRa, MRb
MI32TOF32 MRa, MRb
MI32TOF32 MRa, mem32
MUI32TOF32 MRa, MRb
MUI32TOF32 MRa, mem32



Copyright © 2024 Texas Instruments Incorporated

**MFRACF32 MRa, MRb**

***Fractional Portion of a 32-Bit Floating-Point Value***

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1110 0000 0000
```

**Description**

Returns in MRa the fractional portion of the 32-bit floating-point value in MRb

**Flags**

This instruction does not affect any flags:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.

**Example**

```
MMOVIZ    MR2, #19.625 ; MR2 = 19.625 (0x419D0000)
MFRACF32  MR3, MR2     ; MR3 = MFRACF32(MR2) = 0.625 (0x3F200000)0)
```



Copyright © 2024 Texas Instruments Incorporated

### MI16TOF32 MRa, MRb

***Convert 16-Bit Integer to 32-Bit Floating-Point Value***

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1110 1000 0000
```

**Description**

Convert the 16-bit signed integer in MRb to a 32-bit floating-point value and store the result in MRa.

```
MRa = MI16TOF32(MRb);
```

**Flags**

This instruction does not affect any flags:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.

**Example**

```
MMOVIZ      MR0, #0x0000    ; MR0(31:16) = 0.0 (0x0000)
MMOVXI      MR0, #0x0004    ; MR0(15:0) = 4.0 (0x0004)
MI16TOF32   MR1, MR0        ; MR1 = MI16TOF32 (MR0) = 4.0 (0x40800000)
MMOVIZ      MR2, #0x0000    ; MR2(31:16) = 0.0 (0x0000)
MMOVXI      MR2, #0xFFFC    ; MR2(15:0) = -4.0 (0xFFFC)
MI16TOF32   MR3, MR2        ; MR3 = MI16TOF32 (MR2) = -4.0 (0xC0800000)
MSTOP
```

**See also**

MF32TOI16 MRa, MRb
MF32TOI16R MRa, MRb
MF32TOUI16 MRa, MRb
MF32TOUI16R MRa, MRb
MI16TOF32 MRa, mem16
MUI16TOF32 MRa, mem16
MUI16TOF32 MRa, MRb



Copyright © 2024 Texas Instruments Incorporated

**MI16TOF32 MRa, mem16**

***Convert 16-Bit Integer to 32-Bit Floating-Point Value***

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| mem16 | 16-bit source memory location to be converted |

**Opcode**

```
LSW: mmmm mmmm mmmm mmmm
MSW: 0111 0101 00aa addr
```

**Description**

Convert the 16-bit signed integer indicated by the mem16 pointer to a 32-bit floating-point value and store the result in MRa.

```
MRa = MI16TOF32[mem16];
```

**Flags**

This instruction does not affect any flags:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction:

**Example**

```
; Assume A = 4 (0x0004)
;        B = -4 (0xFFFC)
   MI16TOF32 MR0, @_A ; MR0 = MI16TOF32(A) = 4.0 (0x40800000)
   MI16TOF32 MR1, @_B ; MR1 = MI16TOF32(B) = -4.0 (0xC0800000
```

**See also**

MF32TOI16 MRa, MRb
MF32TOI16R MRa, MRb
MF32TOUI16 MRa, MRb
MF32TOUI16R MRa, MRb
MI16TOF32 MRa, MRb
MUI16TOF32 MRa, mem16
MUI16TOF32 MRa, MRb



Copyright © 2024 Texas Instruments Incorporated

**MI32TOF32 MRa, mem32**

***Convert 32-Bit Integer to 32-Bit Floating-Point Value***

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| mem32 | 32-bit memory source for the MMOV32 operation. |

**Opcode**

```
LSW: mmmm mmmm mmmm mmmm
MSW: 0111 0100 01aa addr
```

**Description**

Convert the 32-bit signed integer indicated by mem32 to a 32-bit floating-point value and store the result in MRa.

```
MRa = MI32TOF32[mem32];
```

**Flags**

This instruction does not affect any flags:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Given X, M and B are IQ24 numbers:
; X = IQ24(+2.5) = 0x02800000
; M = IQ24(+1.5) = 0x01800000
; B = IQ24(-0.5) = 0xFF800000
;
; Calculate Y = X * M + B
;
; Convert M, X, and B from IQ24 to float
;
_Cla1Task3:
   MI32TOF32 MR0, @_M          ; MR0 = 0x4BC00000
   MI32TOF32 MR1, @_X          ; MR1 = 0x4C200000
   MI32TOF32 MR2, @_B          ; MR2 = 0xCB000000
   MMPYF32 MR0, MR0, #0x3380   ; M = 1/(1*2^24) * iqm = 1.5 (0x3FC00000)
   MMPYF32 MR1, MR1, #0x3380   ; X = 1/(1*2^24) * iqx = 2.5 (0x40200000)
   MMPYF32 MR2, MR2, #0x3380   ; B = 1/(1*2^24) * iqb = -.5 (0xBF000000)
   MMPYF32 MR3, MR0, MR1       ; M*X
   MADDF32 MR2, MR2, MR3       ; Y=MX+B = 3.25 (0x40500000)
; Convert Y from float32 to IQ24
   MMPYF32 MR2, MR2, #0x4B80   ; Y * 1*2^24
   MF32TOI32 MR2, MR2          ; IQ24(Y) = 0x03400000
   MMOV32 @_Y, MR2             ; store result
   MSTOP                       ; end of task
```

**See also**

MF32TOI32 MRa, MRb
MF32TOUI32 MRa, MRb
MI32TOF32 MRa, MRb
MUI32TOF32 MRa, MRb
MUI32TOF32 MRa, mem32



Copyright © 2024 Texas Instruments Incorporated

**MI32TOF32 MRa, MRb**

***Convert 32-Bit Integer to 32-Bit Floating-Point Value***

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1101 1000 0000
```

**Description**

Convert the signed 32-bit integer in MRb to a 32-bit floating-point value and store the result in MRa.

```
MRa = MI32TOF32(MRb);
```

**Flags**

This instruction does not affect any flags:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.

**Example**

```
   MMOVIZ    MR2, #0x1111 ; MR2(31:16) = 4369 (0x1111)
   MMOVXI    MR2, #0x1111 ; MR2(15:0) = 4369 (0x1111)
                          ; MR2 = +286331153 (0x11111111)
   MI32TOF32 MR3, MR2     ; MR3 = MI32TOF32 (MR2) = 286331153.0 (0x4D888888)
```

**See also**

MF32TOI32 MRa, MRb
MF32TOUI32 MRa, MRb
MI32TOF32 MRa, mem32
MUI32TOF32 MRa, MRb
MUI32TOF32 MRa, mem32



Copyright © 2024 Texas Instruments Incorporated

## MLSL32 MRa, #SHIFT

### *Logical Shift Left*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point source/destination register (MR0 to MR3) |
| #SHIFT | Number of bits to shift (1 to 32) |

**Opcode**

```
LSW: 0000 0000 0shi ftaa
MSW: 0111 1011 1100 0000
```

**Description**

Logical shift-left of MRa by the number of bits indicated. The number of bits can be 1 to 32.

```
MARa(31:0) = Logical Shift Left(MARa(31:0) by #SHIFT bits);
```

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | No | No |

The MSTF register flags are modified based on the integer results of the operation.

```
NF = MRa(31);
ZF = 0;
if(MRa(31:0) == 0) { ZF = 1; }
```

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Given m2 = (int32)32
;       x2 = (int32)64
;       b2 = (int32)-128
;
; Calculate:
;       m2 = m2*2
;       x2 = x2*4
;       b2 = b2*8
;
_Cla1Task3:
   MMOV32 MR0, @_m2    ; MR0 = 32 (0x00000020)
   MMOV32 MR1, @_x2    ; MR1 = 64 (0x00000040)
   MMOV32 MR2, @_b2    ; MR2 = -128 (0xFFFFFF80)
   MLSL32 MR0, #1      ; MR0 = 64 (0x00000040)
   MLSL32 MR1, #2      ; MR1 = 256 (0x00000100)
   MLSL32 MR2, #3      ; MR2 = -1024 (0xFFFFFC00)
   MMOV32 @_m2, MR0    ; Store results
   MMOV32 @_x2, MR1
   MMOV32 @_b2, MR2
   MSTOP               ; end of task
```



Copyright © 2024 Texas Instruments Incorporated

**MLSL32 MRa, #SHIFT** (continued)

***Logical Shift Left***

**See also**

MADD32 MRa, MRb, MRc
MASR32 MRa, #SHIFT
MAND32 MRa, MRb, MRc
MLSR32 MRa, #SHIFT
MOR32 MRa, MRb, MRc
MXOR32 MRa, MRb, MRc
MSUB32 MRa, MRb, MRc



Copyright © 2024 Texas Instruments Incorporated

## MLSR32 MRa, #SHIFT

### *Logical Shift Right*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point source/destination register (MR0 to MR3) |
| #SHIFT | Number of bits to shift (1 to 32) |

**Opcode**

```
LSW: 0000 0000 0shi ftaa
MSW: 0111 1011 1000 0000
```

**Description**

Logical shift-right of MRa by the number of bits indicated. The number of bits can be 1 to 32. Unlike the arithmetic shift (MASR32), the logical shift does not preserve the number's sign bit. Every bit in the operand is moved the specified number of bit positions, and the vacant bit positions are filled in with zeros.

```
MARa(31:0) = Logical Shift Right(MARa(31:0) by #SHIFT bits);
```

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | No | No |

The MSTF register flags are modified based on the integer results of the operation.

```
NF = MRa(31);
ZF = 0;
if(MRa(31:0) == 0) { ZF = 1;}
```

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Illustrate the difference between MASR32 and MLSR32
MMOVIZ MR0, #0xAAAA   ; MR0 = 0xAAAA5555
MMOVXI MR0, #0x5555
MMOV32 MR1, MR0       ; MR1 = 0xAAAA5555
MMOV32 MR2, MR0       ; MR2 = 0xAAAA5555
MASR32 MR1, #1        ; MR1 = 0xD5552AAA
MLSR32 MR2, #1        ; MR2 = 0x55552AAA
MASR32 MR1, #1        ; MR1 = 0xEAAA9555
MLSR32 MR2, #1        ; MR2 = 0x2AAA9555
MASR32 MR1, #6        ; MR1 = 0xFFAAAA55
MLSR32 MR2, #6        ; MR2 = 0x00AAAA55
```

**See also**

MADD32 MRa, MRb, MRc
MASR32 MRa, #SHIFT
MAND32 MRa, MRb, MRc
MLSL32 MRa, #SHIFT
MOR32 MRa, MRb, MRc
MXOR32 MRa, MRb, MRc
MSUB32 MRa, MRb, MRc



Copyright © 2024 Texas Instruments Incorporated

**MMACF32 MR3, MR2, MRd, MRe, MRf ||MMOV32 MRa, mem32**

***32-Bit Floating-Point Multiply and Accumulate with Parallel Move***

## Operands

| | |
|---|---|
| MR3 | floating-point destination/source register MR3 for the add operation |
| MR2 | CLA floating-point source register MR2 for the add operation |
| MRd | CLA floating-point destination register (MR0 to MR3) for the multiply operation<br>MRd cannot be the same register as MRa |
| MRe | CLA floating-point source register (MR0 to MR3) for the multiply operation |
| MRf | CLA floating-point source register (MR0 to MR3) for the multiply operation |
| MRa | CLA floating-point destination register for the MMOV32 operation (MR0 to MR3).<br>MRa cannot be MR3 or the same register as MRd. |
| mem32 | 32-bit source for the MMOV32 operation |

## Opcode

```
LSW: mmmm mmmm mmmm mmmm
MSW: 0011 ffee ddaa addr
```

## Description

Multiply and accumulate the contents of floating-point registers and move from register to memory. The destination register for the MMOV32 cannot be the same as the destination registers for the MMACF32.

```
MR3 = MR3 + MR2;
MRd = MRe * MRf;
MRa = [mem32];
```

## Restrictions

The destination registers for the MMACF32 and the MMOV32 must be unique. That is, MRa cannot be MR3 and MRa cannot be the same register as MRd.

## Flags

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | Yes | Yes |

The MSTF register flags are modified as follows:

- LUF = 1 if MMACF32 (add or multiply) generates an underflow condition.
- LVF = 1 if MMACF32 (add or multiply) generates an overflow condition.

MMOV32 sets the NF and ZF flags as follows:

```
NF = MRa(31);
ZF = 0;
if(MRa(30:23) == 0) { ZF = 1; NF = 0; }
```

## Pipeline

MMACF32 and MMOV32 complete in a single cycle.



Copyright © 2024 Texas Instruments Incorporated

**MMACF32 MR3, MR2, MRd, MRe, MRf ||MMOV32 MRa, mem32** (continued)

### *32-Bit Floating-Point Multiply and Accumulate with Parallel Move*

**Example 1**

```
; Perform 5 multiply and accumulate operations:
;
; X and Y are 32-bit floating-point arrays
;
; 1st multiply: A = X0 * Y0
; 2nd multiply: B = X1 * Y1
; 3rd multiply: C = X2 * Y2
; 4th multiply: D = X3 * Y3
; 5th multiply: E = X3 * Y3
;
; Result = A + B + C + D + E
;
_Cla1Task1:
   MMOVI16 MAR0, #_X               ; MAR0 points to X array
   MMOVI16 MAR1, #_Y               ; MAR1 points to Y array
   MNOP                            ; Delay for MAR0, MAR1 load
   MNOP                            ; Delay for MAR0, MAR1 load
                                   ; <-- MAR0 valid
   MMOV32 MR0, *MAR0[2]++          ; MR0 = X0, MAR0 += 2
                                   ; <-- MAR1 valid
   MMOV32 MR1, *MAR1[2]++          ; MR1 = Y0, MAR1 += 2
   MMPYF32 MR2, MR0, MR1           ; MR2 = A = X0 * Y0
|| MMOV32 MR0, *MAR0[2]++          ; In parallel MR0 = X1, MAR0 += 2
   MMOV32 MR1, *MAR1[2]++          ; MR1 = Y1, MAR1 += 2
   MMPYF32 MR3, MR0, MR1           ; MR3 = B = X1 * Y1
|| MMOV32 MR0, *MAR0[2]++          ; In parallel MR0 = X2, MAR0 += 2
   MMOV32 MR1, *MAR1[2]++          ; MR1 = Y2, MAR2 += 2
   MMACF32 MR3, MR2, MR2, MR0, MR1 ; MR3 = A + B, MR2 = C = X2 * Y2
|| MMOV32 MR0, *MAR0[2]++          ; In parallel MR0 = X3
   MMOV32 MR1, *MAR1[2]++          ; MR1 = Y3 M
   MACF32 MR3, MR2, MR2, MR0, MR1  ; MR3 = (A + B) + C, MR2 = D = X3 * Y3
|| MMOV32 MR0, *MAR0               ; In parallel MR0 = X4
   MMOV32 MR1, *MAR1               ; MR1 = Y4
   MMPYF32 MR2, MR0, MR1           ; MR2 = E = X4 * Y4
|| MADDF32 MR3, MR3, MR2           ; in parallel MR3 = (A + B + C) + D
   MADDF32 MR3, MR3, MR2           ; MR3 = (A + B + C + D) + E
   MMOV32 @_Result, MR3            ; Store the result
   MSTOP                           ; end of task
```



Copyright © 2024 Texas Instruments Incorporated

**MMACF32 MR3, MR2, MRd, MRe, MRf ||MMOV32 MRa, mem32** (continued)

### *32-Bit Floating-Point Multiply and Accumulate with Parallel Move*

**Example 2**

```
; sum = X0*B0 + X1*B1 + X2*B2 + Y1*A1 + Y2*B2
;
;       X2 = X1
;       X1 = X0
;       Y2 = Y1 ; Y1 = sum
;
_ClaTask2:
     MMOV32     MR0, @_B2       ; MR0 = B2
     MMOV32     MR1, @_X2       ; MR1 = X2
     MMPYF32    MR2, MR1, MR0   ; MR2 = X2*B2
  || MMOV32     MR0, @_B1       ; MR0 = B1
     MMOVD32    MR1, @_X1       ; MR1 = X1, X2 = X1
     MMPYF32    MR3, MR1, MR0   ; MR3 = X1*B1
  || MMOV32     MR0, @_B0       ; MR0 = B0
     MMOVD32    MR1, @_X0       ; MR1 = X0, X1 = X0
;  MR3 = X1*B1 + X2*B2, MR2 = X0*B0
;  MR0 = A2
     MMACF32 MR3, MR2, MR2, MR1, MR0
  || MMOV32 MR0, @_A2 M

     MOV32 MR1, @_Y2            ; MR1 = Y2
;  MR3 = X0*B0 + X1*B1 + X2*B2, MR2 = Y2*A2
;  MR0 = A1
     MMACF32 MR3, MR2, MR2, MR1, MR0
  || MMOV32 MR0, @_A1
     MMOVD32 MR1,@_Y1           ; MR1 = Y1, Y2 = Y1
     MADDF32 MR3, MR3, MR2      ; MR3 = Y2*A2 + X0*B0 + X1*B1 + X2*B2
  || MMPYF32 MR2, MR1, MR0      ; MR2 = Y1*A1
     MADDF32 MR3, MR3, MR2      ; MR3 = Y1*A1 + Y2*A2 + X0*B0 + X1*B1 + X2*B2
     MMOV32 @_Y1, MR3           ; Y1 = MR3
     MSTOP                      ; end of task
```

**See also**

MMPYF32 MRa, MRb, MRc || MADDF32 MRd, MRe, MRf



Copyright © 2024 Texas Instruments Incorporated

**MMAXF32 MRa, MRb**

***32-Bit Floating-Point Maximum***

**Operands**

| | |
|---|---|
| MRa | CLA floating-point source/destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1101 0010 0000
```

**Description**

```
if(MRa < MRb) MRa = MRb;
```

Special cases for the output from the MMAXF32 operation:

- NaN output is converted to infinity
- A denormalized output is converted to positive zero.

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | No | No |

The ZF and NF flags are configured on the result of the operation, not the result stored in the destination register.

```
if(MRa == MRb)  {ZF=1; NF=0;}
if(MRa > MRb) {ZF=0; NF=0;}
if(MRa < MRb) {ZF=0; NF=1;}
```

**Pipeline**

This is a single-cycle instruction.

**Example 1**

```
MMOVIZ   MR0, #5.0  ; MR0 =  5.0  (0x40A00000)
MMOVIZ   MR1, #-2.0 ; MR1 = -2.0  (0xC0000000)
MMOVIZ   MR2, #-1.5 ; MR2 = -1.5  (0xBFC00000)
MMAXF32   MR2, MR1  ; MR2 = -1.5, ZF = NF = 0
MMAXF32   MR1, MR2  ; MR1 = -1.5, ZF = 0, NF = 1
MMAXF32   MR2, MR0  ; MR2 =  5.0, ZF = 0, NF = 1
MAXF32   MR0, MR2   ; MR2 =  5.0, ZF = 1, NF = 0
```



Copyright © 2024 Texas Instruments Incorporated

**MMAXF32 MRa, MRb** (continued)

### *32-Bit Floating-Point Maximum*

**Example 2**

```
; X is an array of 32-bit floating-point values
; Find the maximum value in an array X
; and store the value in Result
;
_Cla1Task1:
  MMOVI16    MAR1,#_X          ; Start address
  MUI16TOF32 MR0, @_len        ; Length of the array
  MNOP                         ; delay for MAR1 load
  MNOP                         ; delay for MAR1 load
  MMOV32     MR1, *MAR1[2]++   ; MR1 = X0
LOOP
  MMOV32     MR2, *MAR1[2]++   ; MR2 = next element
  MMAXF32    MR1, MR2          ; MR1 = MAX(MR1, MR2)
  MADDF32    MR0, MR0, #-1.0   ; Decrememt the counter
  MCMPF32    MR0 #0.0          ; Set/clear flags for MBCNDD
  MNOP
  MNOP
  MNOP
  MBCNDD     LOOP, NEQ         ; Branch if not equal to zero
  MMOV32     @_Result, MR1     ; Always executed
  MNOP                         ; Always executed
  MNOP                         ; Always executed
  MSTOP                        ; End of task
```

**See also**

MCMPF32 MRa, MRb
MCMPF32 MRa, #16FHi
MMAXF32 MRa, #16FHi
MMINF32 MRa, MRb
MMINF32 MRa, #16FHi



Copyright © 2024 Texas Instruments Incorporated

**MMAXF32 MRa, #16FHi**

***32-Bit Floating-Point Maximum***

## Operands

| | |
|---|---|
| MRa | CLA floating-point source/destination register (MR0 to MR3) |
| #16FHi | A 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0. |

## Opcode

```
LSW: IIII IIII IIII IIII
MSW: 0111 1001 0000 00aa
```

## Description

Compare MRa with the floating-point value represented by the immediate operand. If the immediate value is larger, then load the value into MRa.

```
if(MRa < #16FHi:0) MRa = #16FHi:0;
```

#16FHi is a 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0. This addressing mode is most useful for constants where the lowest 16-bits of the mantissa are 0. Some examples are 2.0 (0x40000000), 4.0 (0x40800000), 0.5 (0x3F000000), and -1.5 (0xBFC00000). The assembler accepts either a hex or float as the immediate value. That is, -1.5 can be represented as #-1.5 or #0xBFC0.

Special cases for the output from the MMAXF32 operation:

- NaN output is converted to infinity
- A denormalized output is converted to positive zero.

## Flags

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | No | No |

The ZF and NF flags are configured on the result of the operation, not the result stored in the destination register.

```
if(MRa == #16FHi:0)  {ZF=1; NF=0;}
if(MRa > #16FHi:0) {ZF=0; NF=0;}
if(MRa < #16FHi:0) {ZF=0; NF=1;}
```

## Pipeline

This is a single-cycle instruction.

## Example

```
MMOVIZ    MR0, #5.0  ; MR0 =  5.0  (0x40A00000)
MMOVIZ    MR1, #4.0  ; MR1 =  4.0  (0x40800000)
MMOVIZ    MR2, #-1.5 ; MR2 = -1.5  (0xBFC00000)
MMAXF32   MR0, #5.5  ; MR0 =  5.5, ZF = 0, NF = 1
MMAXF32   MR1, #2.5  ; MR1 =  4.0, ZF = 0, NF = 0
MMAXF32   MR2, #-1.0 ; MR2 = -1.0, ZF = 0, NF = 1
MMAXF32   MR2, #-1.0 ; MR2 = -1.5, ZF = 1, NF = 0
```



Copyright © 2024 Texas Instruments Incorporated

**MMAXF32 MRa, #16FHi** (continued)

***32-Bit Floating-Point Maximum***

**See also**

MMAXF32 MRa, MRb
MMINF32 MRa, MRb
MMINF32 MRa, #16FHi



Copyright © 2024 Texas Instruments Incorporated

**MMINF32 MRa, MRb**

***32-Bit Floating-Point Minimum***

**Operands**

| | |
|---|---|
| MRa | CLA floating-point source/destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1101 0100 0000
```

**Description**

```
if(MRa > MRb) MRa = MRb;
```

Special cases for the output from the MMINF32 operation:

- NaN output is converted to infinity
- A denormalized output is converted to positive zero.

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | No | No |

The ZF and NF flags are configured on the result of the operation, not the result stored in the destination register.

```
if(MRa == MRb)  {ZF=1; NF=0;}
if(MRa > MRb) {ZF=0; NF=0;}
if(MRa < MRb) {ZF=0; NF=1;}
```

**Pipeline**

This is a single-cycle instruction.

**Example 1**

```
MMOVIZ MR0, #5.0  ; MR0 = 5.0 (0x40A00000)
MMOVIZ MR1, #4.0  ; MR1 = 4.0 (0x40800000)
MMOVIZ MR2, #-1.5 ; MR2 = -1.5 (0xBFC00000)
MMINF32 MR0, MR1  ; MR0 = 4.0, ZF = 0, NF = 0
MMINF32 MR1, MR2  ; MR1 = -1.5, ZF = 0, NF = 0
MMINF32 MR2, MR1  ; MR2 = -1.5, ZF = 1, NF = 0
MMINF32 MR1, MR0  ; MR2 = -1.5, ZF = 0, NF = 1
```



Copyright © 2024 Texas Instruments Incorporated

**MMINF32 MRa, MRb** (continued)

### *32-Bit Floating-Point Minimum*

**Example 2**

```
;
; X is an array of 32-bit floating-point values
; Find the minimum value in an array X
; and store the value in Result
;
_Cla1Task1:
MMOVI16    MAR1,#_X          ; Start address
MUI16TOF32 MR0, @_len        ; Length of the array
MNOP                         ; delay for MAR1 load
MNOP                         ; delay for MAR1 load
MMOV32     MR1, *MAR1[2]++   ; MR1 = X0
LOOP
MMOV32     MR2, *MAR1[2]++   ; MR2 = next element
MMINF32    MR1, MR2          ; MR1 = MAX(MR1, MR2)
MADDF32    MR0, MR0, #-1.0   ; Decrememt the counter
MCMPF32    MR0 #0.0          ; Set/clear flags for MBCNDD
MNOP
MNOP
MNOP
MBCNDD     LOOP, NEQ         ; Branch if not equal to zero
MMOV32     @_Result, MR1     ; Always executed
MNOP                         ; Always executed
MNOP                         ; Always executed
MSTOP                        ; End of task
```

**See also**

MMAXF32 MRa, MRb
MMAXF32 MRa, #16FHi
MMINF32 MRa, #16FHi



Copyright © 2024 Texas Instruments Incorporated

**MMINF32 MRa, #16FHi**

***32-Bit Floating-Point Minimum***

## Operands

| | |
|---|---|
| MRa | floating-point source/destination register (MR0 to MR3) |
| #16FHi | A 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0. |

## Opcode

```
LSW: IIII IIII IIII IIII
MSW: 0111 1001 0100 00aa
```

## Description

Compare MRa with the floating-point value represented by the immediate operand. If the immediate value is smaller, then load the value into MRa.

```
if(MRa > #16FHi:0) MRa = #16FHi:0;
```

#16FHi is a 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0. This addressing mode is most useful for constants where the lowest 16-bits of the mantissa are 0. Some examples are 2.0 (0x40000000), 4.0 (0x40800000), 0.5 (0x3F000000), and -1.5 (0xBFC00000). The assembler accepts either a hex or float as the immediate value. That is, -1.5 can be represented as #-1.5 or #0xBFC0.

Special cases for the output from the MMINF32 operation:

- NaN output is converted to infinity
- A denormalized output is converted to positive zero.

## Flags

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | No | No |

The ZF and NF flags are configured on the result of the operation, not the result stored in the destination register.

```
if(MRa == #16FHi:0)   {ZF=1; NF=0;}
if(MRa > #16FHi:0) {ZF=0; NF=0;}
if(MRa < #16FHi:0) {ZF=0; NF=1;}
```

## Pipeline

This is a single-cycle instruction.

## Example

```
MMOVIZ   MR0, #5.0   ; MR0 =  5.0  (0x40A00000)
MMOVIZ   MR1, #4.0   ; MR1 =  4.0  (0x40800000)
MMOVIZ   MR2, #-1.5  ; MR2 = -1.5  (0xBFC00000)
MMINF32  MR0, #5.5   ; MR0 =  5.0, ZF = 0, NF = 1
MMINF32  MR1, #2.5   ; MR1 =  2.5, ZF = 0, NF = 0
MMINF32  MR2, #-1.0  ; MR2 = -1.5, ZF = 0, NF = 1
MMINF32  MR2, #-1.5  ; MR2 = -1.5, ZF = 1, NF = 0
```



Copyright © 2024 Texas Instruments Incorporated

**MMINF32 MRa, #16FHi** (continued)

***32-Bit Floating-Point Minimum***

**See also**

MMAXF32 MRa, #16FHi
MMAXF32 MRa, MRb
MMINF32 MRa, MRb



Copyright © 2024 Texas Instruments Incorporated

### MMOV16 MARx, MRa, #16I

***Load the Auxiliary Register with MRa + 16-bit Immediate Value***

**Operands**

| | |
|---|---|
| MARx | Auxiliary register MAR0 or MAR1 |
| MRa | CLA Floating-point register (MR0 to MR3) |
| #16I | 16-bit immediate value |

**Opcode**

```
LSW: IIII IIII IIII IIII (opcode of MMOV16 MAR0, MRa, #16I)
MSW: 0111 1111 1101 00AA
LSW: IIII IIII IIII IIII (opcode of MMOV16 MAR1, MRa, #16I)
MSW: 0111 1111 1111 00AA
```

**Description**

Load the auxiliary register, MAR0 or MAR1, with MRa(15:0) + 16-bit immediate value. Refer to the Pipeline section for important information regarding this instruction.

```
MARx = MRa(15:0) + #16I;
```

**Flags**

This instruction does not modify flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction. The load of MAR0 or MAR1 occurs in the EXE phase of the pipeline. Any post increment of MAR0 or MAR1 using indirect addressing occurs in the D2 phase of the pipeline. Therefore, the following applies when loading the auxiliary registers:

- **I1 and I2**

  The two instructions following MMOV16 use MAR0 or MAR1 before the update occurs. Thus, these two instructions use the old value of MAR0 or MAR1.
- **I3**

  Loading of an auxiliary register occurs in the EXE phase while updates due to post-increment addressing occur in the D2 phase. Thus, I3 cannot use the auxiliary register or there is a conflict. In the case of a conflict, the update due to address-mode post increment wins and the auxiliary register is not updated with #_X.
- **I4**

  Starting with the 4th instruction, MAR0 or MAR1 is the new value loaded with MMOVI16.

```
; Assume MAR0 is 50, MR0 is 10, and #_X is 20
  MMOV16 MAR0, MR0, #_X        ; Load MAR0 with address of X (20) + MR0 (10)
  <Instruction 1>         ; I1 Uses the old value of MAR0 (50)
  <Instruction 2>         ; I2 Uses the old value of MAR0 (50)
  <Instruction 3>         ; I3 Cannot use MAR0
  <Instruction 4>         ; I4 Uses the new value of MAR0 (30)
  <Instruction 5>         ; I5
```



Copyright © 2024 Texas Instruments Incorporated

**MMOV16 MARx, MRa, #16I** (continued)

***Load the Auxiliary Register with MRa + 16-bit Immediate Value***

**Table 6-15. Pipeline Activity for MMOV16 MARx, MRa , #16I**

| Instruction | F1 | F2 | D1 | D2 | R1 | R2 | E | W |
|---|---|---|---|---|---|---|---|---|
| MMOV16 MAR0, MR0, #_X | MMOV16 | | | | | | | |
| I1 | I1 | MMOV16 | | | | | | |
| I2 | I2 | I1 | MMOV16 | | | | | |
| I3 | I3 | I2 | I1 | MMOV16 | | | | |
| I4 | I4 | I3 | I2 | I1 | MMOV16 | | | |
| I5 | I5 | I4 | I3 | I2 | I1 | MMOV16 | | |
| I6 | I6 | I5 | I4 | I3 | I2 | I1 | MMOV16 | |

**Example 1**

```
; Calculate an offset into a sin/cos table
;
_Cla1Task1:
    MMOV32 MR0,@_rad                ; MR0 = rad
    MMOV32 MR1,@_TABLE_SIZEDivTwoPi ; MR1 = TABLE_SIZE/(2*Pi)
    MMPYF32 MR1,MR0,MR1             ; MR1 = rad* TABLE_SIZE/(2*Pi)
||  MMOV32 MR2,@_TABLE_MASK         ; MR2 = TABLE_MASK
    MF32TOI32 MR3,MR1               ; MR3 = K=int(rad*TABLE_SIZE/(2*Pi))
    MAND32 MR3,MR3,MR2              ; MR3 = K & TABLE_MASK
    MLSL32 MR3,#1                   ; MR3 = K * 2
    MMOV16 MAR0,MR3,#_Cos0          ; MAR0 K*2+addr of table.Cos0
    MFRACF32 MR1,MR1                ; I1
    MMOV32 MR0,@_TwoPiDivTABLE_SIZE ; I2
    MMPYF32 MR1,MR1,MR0             ; I3
||  MMOV32 MR0,@_Coef3
    MMOV32 MR2,*MAR0[#-64]++        ; MR2 = *MAR0, MAR0 += (-64)
    ...
    ...
    MSTOP ; end of task
```



Copyright © 2024 Texas Instruments Incorporated

**MMOV16 MARx, MRa, #16I** (continued)

***Load the Auxiliary Register with MRa + 16-bit Immediate Value***

**Example 2**

```
; This task logs the last NUM_DATA_POINTS
; ADCRESULT1 values in the array VoltageCLA
;
; When the last element in the array has been
; filled, the task goes back to the
; the first element.
;
; Before starting the ADC conversions, force
; Task 8 to initialize the ConversionCount to zero
;
; The ADC is set to sample (acquire) for 15 SYSCLK cycles
; or 75ns. After the capacitor has captured the analog
; value, the ADC triggers this task early.
; It takes 10.5 ADCCLKs to complete a conversion,
; the ADCCLK being SYSCLK/4
;   T_sys = 1/200MHz = 5ns
;   T_adc = 4*T_sys  = 20ns
; The ADC takes 10.5 * 4 or 42 SYSCLK cycles to complete
; a conversion. The ADC result register can be read on the
; 36th instruction after the task begins.
;
_Cla1Task2:
    .asg       0, N
    .loop
    MNOP                                    ;I1 - I28 Wait till I36 to read
result
    .eval      N + 1, N
    .break     N = 28
    .endloop
    MMOVZ16    MR0, @_ConversionCount           ;I29 Current Conversion
    MMOV16     MAR1, MR0, #_VoltageCLA          ;I30 Next array location
    MUI16TOF32 MR0, MR0                         ;I31 Convert count to float32
    MADDF32    MR0, MR0, #1.0                   ;I32 Add 1 to conversion count
    MCMPF32    MR0, #NUM_DATA_POINTS.0          ;I33 Compare count to max
    MF32TOUI16 MR0, MR0                         ;I34 Convert count to Uint16
    MNOP                                        ;I35 Wait till I36 to read
result
    MMOVZ16    MR2, @_AdcaResultRegs.ADCRESULT1 ;I36 Read ADCRESULT1
    MMOV16     *MAR1, MR2                       ; Store ADCRESULT1
    MBCNDD     _RestartCount, GEQ               ; If count >= NUM_DATA_POINTS
    MMOVIZ     MR1, #0.0                        ; Always executed: MR1=0
    MNOP
    MNOP
    MMOV16     @_ConversionCount, MR0           ; If branch not taken
    MSTOP                                       ; store current count
_RestartCount
    MMOV16     @_ConversionCount, MR1           ; If branch taken, restart
count
    MSTOP                                       ; end of task
; This task initializes the ConversionCount
; to zero
;
_Cla1Task8:
    MMOVIZ     MR0, #0.0
    MMOV16     @_ConversionCount, MR0
    MSTOP
_ClaT8End:
```



Copyright © 2024 Texas Instruments Incorporated

**MMOV16 MARx, mem16**

***Load MAR1 with 16-bit Value***

**Operands**

| | |
|---|---|
| MARx | CLA auxiliary register MAR0 or MAR1 |
| mem16 | 16-bit destination memory accessed using one of the available addressing modes |

**Opcode**

```
LSW: mmmm mmmm mmmm mmmm (Opcode for MMOV16 MAR0, mem16)
MSW: 0111 0110 0000 addr
LSW: mmmm mmmm mmmm mmmm (Opcode for MMOV16 MAR1, mem16)
MSW: 0111 0110 0100 addr
```

**Description**

Load MAR0 or MAR1 with the 16-bit value pointed to by mem16. Refer to the Pipeline section for important information regarding this instruction.

```
MAR1 = [mem16];
```

**Flags**

No flags MSTF flags are affected.

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction. The load of MAR0 or MAR1 occurs in the EXE phase of the pipeline. Any post increment of MAR0 or MAR1 using indirect addressing occurs in the D2 phase of the pipeline. Therefore, the following applies when loading the auxiliary registers:

- **I1 and I2**

  The two instructions following MMOV16 use MAR0 or MAR1 before the update occurs. Thus, these two instructions use the old value of MAR0 or MAR1.
- **I3**

  Loading of an auxiliary register occurs in the EXE phase while updates due to post-increment addressing occur in the D2 phase. Thus, I3 cannot use the auxiliary register or there is a conflict. In the case of a conflict, the update due to address-mode post increment, the auxiliary register is not updated with #_X.
- **I4**

  Starting with the 4th instruction, MAR0 or MAR1 is the new value loaded with MMOV16.

```
; Assume MAR0 is 50 and @_X is 20
MMOV16 MAR0, @_X    ; Load MAR0 with the contents of X (20)
<Instruction 1>     ; I1 Uses the old value of MAR0 (50)
<Instruction 2>     ; I2 Uses the old value of MAR0 (50)
<Instruction 3>     ; I3 Cannot use MAR0
<Instruction 4>     ; I4 Uses the new value of MAR0 (20)
<Instruction 5>     ; I5
....
```



Copyright © 2024 Texas Instruments Incorporated

**MMOV16 MARx, mem16** (continued)

***Load MAR1 with 16-bit Value***

**Table 6-16. Pipeline Activity for MMOV16 MAR0/MAR1, mem16**

| Instruction | F1 | F2 | D1 | D2 | R1 | R2 | E | W |
|---|---|---|---|---|---|---|---|---|
| MMOV16 MAR0, @_X | MMOV16 | | | | | | | |
| I1 | I1 | MMOV16 | | | | | | |
| I2 | I2 | I1 | MMOV16 | | | | | |
| I3 | I3 | I2 | I1 | MMOV16 | | | | |
| I4 | I4 | I3 | I2 | I1 | MMOV16 | | | |
| I5 | I5 | I4 | I3 | I2 | I1 | MMOV16 | | |
| I6 | I6 | I5 | I4 | I3 | I2 | I1 | MMOV16 | |



Copyright © 2024 Texas Instruments Incorporated

**MMOV16 MARx, mem16** (continued)

***Load MAR1 with 16-bit Value***

**Example**

```
; This task logs the last NUM_DATA_POINTS
; ADCRESULT1 values in the array VoltageCLA
;
; When the last element in the array has been
; filled, the task goes back to the
; the first element.
;
; Before starting the ADC conversions, force
; Task 8 to initialize the ConversionCount to zero
;
; The ADC is set to sample (acquire) for 15 SYSCLK cycles
; or 75ns. After the capacitor has captured the analog
; value, the ADC triggers this task early.
; It takes 10.5 ADCCLKs to complete a conversion,
; the ADCCLK being SYSCLK/4
;   T_sys = 1/200MHz = 5ns
;   T_adc = 4*T_sys  = 20ns
; The ADC takes 10.5 * 4 or 42 SYSCLK cycles to complete
; a conversion. The ADC result register can be read on the
; 36th instruction after the task begins.
;
_Cla1Task2:
    .asg       0, N
    .loop
    MNOP
;I1 - I28 Wait till I36 to read result
    .eval      N + 1, N
    .break     N = 28
    .endloop
    MMOVZ16    MR0, @_ConversionCount              ;I29 Current Conversion
    MMOV16     MAR1, MR0, #_VoltageCLA             ;I30 Next array location
    MUI16TOF32 MR0, MR0                            ;I31 Convert count to float32
    MADDF32    MR0, MR0, #1.0                      ;I32 Add 1 to conversion count
    MCMPF32    MR0, #NUM_DATA_POINTS.0             ;I33 Compare count to max
    MF32TOUI16 MR0, MR0                            ;I34 Convert count to Uint16
    MNOP                                           ;I35 Wait until I36 to read
result
    MMOVZ16    MR2, @_AdcaResultRegs.ADCRESULT1    ;I36 Read ADCRESULT1
    MMOV16     *MAR1, MR2                          ; Store ADCRESULT1
    MBCNDD     _RestartCount, GEQ                  ; If count >= NUM_DATA_POINTS
    MMOVIZ     MR1, #0.0                           ; Always executed: MR1=0
    MNOP
    MNOP
    MMOV16     @_ConversionCount, MR0              ; If branch not taken
    MSTOP                                          ; store current count
_RestartCount
    MMOV16     @_ConversionCount, MR1              ; If branch taken, restart
count
    MSTOP                                          ; end of task
; This task initializes the ConversionCount
; to zero
;
_Cla1Task8:
    MMOVIZ     MR0, #0.0
    MMOV16     @_ConversionCount, MR0
    MSTOP
_ClaT8End:
```



Copyright © 2024 Texas Instruments Incorporated

### MMOV16 mem16, MARx

***Move 16-Bit Auxiliary Register Contents to Memory***

**Operands**

| | |
|---|---|
| mem16 | 16-bit destination memory accessed using one of the available addressing modes |
| MARx | CLA auxiliary register MAR0 or MAR1 |

**Opcode**

```
LSW: mmmm mmmm mmmm mmmm (Opcode for MMOV16 mem16, MAR0)
MSW: 0111 0110 1000 addr
LSW: mmmm mmmm mmmm mmmm (Opcode for MMOV16 mem16, MAR1)
MSW: 0111 0110 1100 addr
```

**Description**

Store the contents of MAR0 or MAR1 in the 16-bit memory location pointed to by mem16.

```
[mem16] = MAR0;
```

**Flags**

No flags MSTF flags are affected.

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.



Copyright © 2024 Texas Instruments Incorporated

**MMOV16 mem16, MRa**

***Move 16-Bit Floating-Point Register Contents to Memory***

**Operands**

| | |
|---|---|
| mem16 | 16-bit destination memory accessed using one of the available addressing modes |
| MRa | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: mmmm mmmm mmmm mmmm
MSW: 0111 0101 11aa addr
```

**Description**

Move 16-bit value from the lower 16-bits of the floating-point register (MRa(15:0)) to the location pointed to by mem16.

```
[mem16] = MRa(15:0);
```

**Flags**

No flags MSTF flags are affected.

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.



Copyright © 2024 Texas Instruments Incorporated

**MMOV16 mem16, MRa** (continued)

***Move 16-Bit Floating-Point Register Contents to Memory***

**Example**

```
; This task logs the last NUM_DATA_POINTS
; ADCRESULT1 values in the array VoltageCLA
;
; When the last element in the array has been
; filled, the task goes back to the
; the first element.
;
; Before starting the ADC conversions, force
; Task 8 to initialize the ConversionCount to zero
;
; The ADC is set to sample (acquire) for 15 SYSCLK cycles
; or 75ns. After the capacitor has captured the analog
; value, the ADC triggers this task early.
; It takes 10.5 ADCCLKs to complete a conversion,
; the ADCCLK being SYSCLK/4
;   T_sys = 1/200MHz = 5ns
;   T_adc = 4*T_sys  = 20ns
; The ADC takes 10.5 * 4 or 42 SYSCLK cycles to complete
; a conversion. The ADC result register can be read on the
; 36th instruction after the task begins.
;
_Cla1Task2:
    .asg       0, N
    .loop
    MNOP
;I1 - I28 Wait till I36 to read result
    .eval      N + 1, N
    .break     N = 28
    .endloop
    MMOVZ16    MR0, @_ConversionCount              ;I29 Current Conversion
    MMOV16     MAR1, MR0, #_VoltageCLA             ;I30 Next array location
    MUI16TOF32 MR0, MR0                            ;I31 Convert count to float32
    MADDF32    MR0, MR0, #1.0                      ;I32 Add 1 to conversion count
    MCMPF32    MR0, #NUM_DATA_POINTS.0             ;I33 Compare count to max
    MF32TOUI16 MR0, MR0                            ;I34 Convert count to Uint16
    MNOP                                           ;I35 Wait till I36 to read
result
    MMOVZ16    MR2, @_AdcaResultRegs.ADCRESULT1    ;I36 Read ADCRESULT1
    MMOV16     *MAR1, MR2                          ; Store ADCRESULT1
    MBCNDD     _RestartCount, GEQ                  ; If count >= NUM_DATA_POINTS
    MMOVIZ     MR1, #0.0                           ; Always executed: MR1=0
    MNOP
    MNOP
    MMOV16     @_ConversionCount, MR0              ; If branch not taken
    MSTOP                                          ; store current count
_RestartCount
    MMOV16     @_ConversionCount, MR1              ; If branch taken, restart
count
    MSTOP                                          ; end of task
; This task initializes the ConversionCount
; to zero
;
_Cla1Task8:
    MMOVIZ     MR0, #0.0
    MMOV16     @_ConversionCount, MR0
    MSTOP
_ClaT8End:
```

**See also**

MMOVIZ MRa, #16FHi
MMOVXI MRa, #16FLoHex



Copyright © 2024 Texas Instruments Incorporated

**MMOV32 mem32, MRa**

***Move 32-Bit Floating-Point Register Contents to Memory***

**Operands**

| | |
|---|---|
| MRa | floating-point register (MR0 to MR3) |
| mem32 | 32-bit destination memory accessed using one of the available addressing modes |

**Opcode**

```
LSW: mmmm mmmm mmmm mmmm
MSW: 0111 0100 11aa addr
```

**Description**

Move from MRa to 32-bit memory location indicated by mem32.

```
[mem32] = MRa;
```

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

No flags affected.

**Pipeline**

This is a single-cycle instruction.



Copyright © 2024 Texas Instruments Incorporated

**MMOV32 mem32, MRa** (continued)

***Move 32-Bit Floating-Point Register Contents to Memory***

**Example**

```
; Perform 5 multiply and accumulate operations:
;
; X and Y are 32-bit floating-point arrays;
; 1st multiply: A = X0 * Y0
; 2nd multiply: B = X1 * Y1
; 3rd multiply: C = X2 * Y2
; 4th multiply: D = X3 * Y3
; 5th multiply: E = X3 * Y3;
; Result = A + B + C + D + E
;
_Cla1Task1:
   MMOVI16       MAR0, #_X               ; MAR0 points to X array
   MMOVI16       MAR1, #_Y               ; MAR1 points to Y array
   MNOP                                  ; Delay for MAR0, MAR1 load
   MNOP                                  ; Delay for MAR0, MAR1 load
                                         ; <-- MAR0 valid
   MMOV32        MR0, *MAR0[2]++         ; MR0 = X0, MAR0 += 2
                                         ; <-- MAR1 valid
   MMOV32        MR1, *MAR1[2]++         ; MR1 = Y0, MAR1 += 2
   MMPYF32       MR2,  MR0, MR1          ; MR2 = A = X0 * Y0
|| MMOV32        MR0, *MAR0[2]++         ; In parallel MR0 = X1, MAR0 += 2
   MMOV32        MR1, *MAR1[2]++         ; MR1 = Y1, MAR1 += 2
   MMPYF32       MR3, MR0, MR1           ; MR3 = B = X1 * Y1
|| MMOV32        MR0, *MAR0[2]++         ; In parallel MR0 = X2, MAR0 += 2
   MMOV32        MR1, *MAR1[2]++         ; MR1 = Y2, MAR2 += 2
   MMACF32       MR3, MR2, MR2, MR0, MR1 ; MR3 = A + B, MR2 = C = X2 * Y2
|| MMOV32        MR0, *MAR0[2]++         ; In parallel MR0 = X3
   MMOV32        MR1, *MAR1[2]++         ; MR1 = Y3
   MMACF32       MR3, MR2, MR2, MR0, MR1 ; MR3 = (A + B) + C, MR2 = D = X3 *
Y3
|| MMOV32        MR0, *MAR0              ; In parallel MR0 = X4
   MMOV32        MR1, *MAR1              ; MR1 = Y4
   MMPYF32       MR2, MR0, MR1           ; MR2 = E = X4 * Y4
|| MADDF32       MR3, MR3, MR2           ; in parallel MR3 = (A + B + C) + D
   MADDF32       MR3, MR3, MR2           ; MR3 = (A + B + C + D) + E
   MMOV32        @_Result, MR3           ; Store the result MSTOP ; end of task
```

**See also**

MMOV32 mem32, MSTF



Copyright © 2024 Texas Instruments Incorporated

**MMOV32 mem32, MSTF**

***Move 32-Bit MSTF Register to Memory***

**Operands**

| | |
|---|---|
| MSTF | Floating-point status register |
| mem32 | 32-bit destination memory |

**Opcode**

```
LSW: mmmm mmmm mmmm mmmm
MSW: 0111 0111 0100 addr
```

**Description**

Copy the CLA floating-point status register, MSTF, to memory.

```
[mem32] = MSTF;
```

**Flags**

This instruction does not modify flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.

One of the uses of this instruction is to save off the return PC (RPC) prior to calling a function. The decision to jump to a function is made when the MCCNDD is in the decode2 (D2) phase of the pipeline; the RPC is also updated in this phase. The actual jump occurs 3 cycles later when MCCNDD enters the execution (E) phase. You must save the old RPC before MCCNDD updates in the D2 phase; that is, save MSTF 3 instructions prior to the function call.

**Example**

The following example illustrates the pipeline flow for the context save (of the flags and RPC) prior to a function call. The first column in the comments shows the pipeline stages for the MMOV32 instruction while the second column pertains to the MCCNDD instruction.

```
 MMOV32 @_temp, MSTF ; D2|   |
MNOP                 ; R1|F1| MCCNDD is fetched
MNOP                 ; R2|F2|
MNOP                 ; E |D1|
MCCNDD _bar, UNC     ; W |D2| old RPC written to memory,
                     ;   |  | RPC updated with MPC+1
MNOP                 ;   |R1|
MNOP                 ;   |R2|
MNOP                 ;   |E | execution branches to _bar
```

**See also**

MMOV32 mem32, MRa



Copyright © 2024 Texas Instruments Incorporated

**MMOV32 MRa, mem32 {, CNDF}**

***Conditional 32-Bit Move***

## Operands

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| mem32 | 32-bit memory location accessed using one of the available addressing modes |
| CNDF | Optional condition |

## Opcode

```
LSW: mmmm mmmm mmmm mmmm
MSW: 0111 00cn dfaa addr
```

## Description

If the condition is true, then move the 32-bit value referenced by mem32 to the floating-point register indicated by MRa.

```
if (CNDF == TRUE) MRa = [mem32];
```

CNDF is one of the following conditions:

| Encode[1] | CNDF | Description | MSTF Flags Tested |
|---|---|---|---|
| 0000 | NEQ | Not equal to zero | ZF == 0 |
| 0001 | EQ | Equal to zero | ZF == 1 |
| 0010 | GT | Greater than zero | ZF == 0 AND NF == 0 |
| 0011 | GEQ | Greater than or equal to zero | NF == 0 |
| 0100 | LT | Less than zero | NF == 1 |
| 0101 | LEQ | Less than or equal to zero | ZF == 1 OR NF == 1 |
| 1010 | TF | Test flag set | TF == 1 |
| 1011 | NTF | Test flag not set | TF == 0 |
| 1100 | LU | Latched underflow | LUF == 1 |
| 1101 | LV | Latched overflow | LVF == 1 |
| 1110 | UNC | Unconditional | None |
| 1111 | UNCF[2] | Unconditional with flag modification | None |

(1) Values not shown are reserved.
(2) This is the default operation, if no CNDF field is specified. This condition allows the ZF and NF flags to be modified when a conditional operation is executed. All other conditions do not modify these flags.

## Flags

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | No | No |

```
if(CNDF == UNCF)
{
  NF = MRa(31);
  ZF = 0;
  if(MRa(30:23) == 0) { ZF = 1; NF = 0; }
}
  else No flags modified;
```



Copyright © 2024 Texas Instruments Incorporated

**MMOV32 MRa, mem32 {, CNDF}** (continued)

***Conditional 32-Bit Move***

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Given A, B, X, M1 and M2 are 32-bit floating-point numbers
;
; if(A == B) calculate Y = X*M1
; if(A! = B) calculate Y = X*M2
;
_Cla1Task5:
    MMOV32    MR0, @_A
    MMOV32    MR1, @_B
    MCMPF32   MR0, MR1
    MMOV32    MR2, @_M1, EQ  ; if A == B, MR2 = M1
                             ;     Y = M1*X
    MMOV32    MR2, @_M2, NEQ ; if A! = B, MR2 = M2
                             ;     Y = M2*X
    MMOV32    MR3, @_X
    MMPYF32   MR3, MR2, MR3  ; Calculate Y
    MMOV32    @_Y, MR3       ; Store Y
    MSTOP                    ; end of task
```

**See also**

MMOV32 MRa, MRb {, CNDF}
MMOVD32 MRa, mem32



Copyright © 2024 Texas Instruments Incorporated

**MMOV32 MRa, MRb {, CNDF}**

***Conditional 32-Bit Move***

## Operands

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |
| CNDF | Optional condition |

## Opcode

```
LSW: 0000 0000 cndf bbaa
MSW: 0111 1010 1100 0000
```

## Description

If the condition is true, then move the 32-bit value in MRb to the floating-point register indicated by MRa.

```
if (CNDF == TRUE) MRa = MRb;
```

CNDF is one of the following conditions:

| Encode(1) | CNDF | Description | MSTF Flags Tested |
|---|---|---|---|
| 0000 | NEQ | Not equal to zero | ZF == 0 |
| 0001 | EQ | Equal to zero | ZF == 1 |
| 0010 | GT | Greater than zero | ZF == 0 AND NF == 0 |
| 0011 | GEQ | Greater than or equal to zero | NF == 0 |
| 0100 | LT | Less than zero | NF == 1 |
| 0101 | LEQ | Less than or equal to zero | ZF == 1 OR NF == 1 |
| 1010 | TF | Test flag set | TF == 1 |
| 1011 | NTF | Test flag not set | TF == 0 |
| 1100 | LU | Latched underflow | LUF == 1 |
| 1101 | LV | Latched overflow | LVF == 1 |
| 1110 | UNC | Unconditional | None |
| 1111 | UNCF(2) | Unconditional with flag modification | None |

(1) Values not shown are reserved.
(2) This is the default operation, if no CNDF field is specified. This condition allows the ZF, and NF flags to be modified when a conditional operation is executed. All other conditions do not modify these flags.

## Flags

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | No | No |

```
if(CNDF == UNCF)
 {
   NF = MRa(31); ZF = 0;
   if(MRa(30:23) == 0) {ZF = 1; NF = 0;}
}
else No flags modified;
```



Copyright © 2024 Texas Instruments Incorporated

**MMOV32 MRa, MRb {, CNDF}** (continued)

***Conditional 32-Bit Move***

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Given: X = 8.0
;        Y = 7.0
;        A = 2.0
;        B = 5.0
; _ClaTask1
    MMOV32   MR3, @_X      ; MR3 = X = 8.0
    MMOV32   MR0, @_Y      ; MR0 = Y = 7.0
    MMAXF32  MR3, MR0      ; ZF = 0, NF = 0, MR3 = 8.0
    MMOV32   MR1, @_A, GT  ; true, MR1 = A = 2.0
    MMOV32   MR1, @_B, LT  ; false, does not load MR1
    MMOV32   MR2, MR1, GT  ; true, MR2 = MR1 = 2.0
    MMOV32   MR2, MR0, LT  ; false, does not load MR2
    MSTOP
```

**See also**

MMOV32 MRa, mem32 {,CNDF}



Copyright © 2024 Texas Instruments Incorporated

**MMOV32 MSTF, mem32**

***Move 32-Bit Value from Memory to the MSTF Register***

**Operands**

| | |
|---|---|
| MSTF | CLA status register |
| mem32 | 32-bit source memory location |

**Opcode**

```
LSW: mmmm mmmm mmmm mmmm
MSW: 0111 0111 0000 addr
```

**Description**

Move from memory to the CLA's status register MSTF. This instruction is most useful when nesting function calls (using MCCNDD).

```
MSTF = [mem32];
```

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | Yes | Yes | Yes | Yes | Yes |

Loading the status register can overwrite all flags and the RPC field. The MEALLOW field is not affected.

**Pipeline**

This is a single-cycle instruction.

**See also**

MMOV32 mem32, MSTF



Copyright © 2024 Texas Instruments Incorporated

**MMOVD32 MRa, mem32**

***Move 32-Bit Value from Memory with Data Copy***

**Operands**

| | |
|---|---|
| MRa | CLA floating-point register (MR0 to MR3) |
| mem32 | 32-bit memory location accessed using one of the available addressing modes |

**Opcode**

```
LSW: mmmm mmmm mmmm mmmm
MSW: 0111 0100 00aa addr
```

**Description**

Move the 32-bit value referenced by mem32 to the floating-point register indicated by MRa.

```
MRa = [mem32];
[mem32+2] = [mem32];
```

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | No | No |

```
NF = MRa(31);
ZF = 0;
if(MRa(30:23) == 0){ ZF = 1; NF = 0; }
```

**Pipeline**

This is a single-cycle instruction.



Copyright © 2024 Texas Instruments Incorporated

**MMOVD32 MRa, mem32** (continued)

***Move 32-Bit Value from Memory with Data Copy***

**Example**

```
; sum = X0*B0 + X1*B1 + X2*B2 + Y1*A1 + Y2*B2
;
;       X2 = X1
;       X1 = X0
;       Y2 = Y1
;       Y1 = sum
;
_Cla1Task2:
    MMOV32 MR0, @_B2        ; MR0 = B2
    MMOV32 MR1, @_X2        ; MR1 = X2
    MMPYF32 MR2, MR1, MR0   ; MR2 = X2*B2
||  MMOV32 MR0, @_B1        ; MR0 = B1
    MMOVD32 MR1, @_X1       ; MR1 = X1, X2 = X1
    MMPYF32 MR3, MR1, MR0   ; MR3 = X1*B1
||  MMOV32 MR0, @_B0        ; MR0 = B0
    MMOVD32 MR1, @_X0       ; MR1 = X0, X1 = X0
; MR3 = X1*B1 + X2*B2, MR2 = X0*B0
; MR0 = A2
    MMACF32 MR3, MR2, MR2, MR1, MR0
||  MMOV32 MR0, @_A2

    MMOV32 MR1, @_Y2        ; MR1 = Y2
; MR3 = X0*B0 + X1*B1 + X2*B2, MR2 = Y2*A2
; MR0 = A1
    MMACF32 MR3, MR2, MR2, MR1, MR0
||  MMOV32 MR0, @_A1
    MMOVD32 MR1,@_Y1        ; MR1 = Y1, Y2 = Y1
    MADDF32 MR3, MR3, MR2   ; MR3 = Y2*A2 + X0*B0 + X1*B1 + X2*B2
||  MMPYF32 MR2, MR1, MR0   ; MR2 = Y1*A1
    MADDF32 MR3, MR3, MR2   ; MR3 = Y1*A1 + Y2*A2 + X0*B0 + X1*B1 + X2*B2
    MMOV32 @_Y1, MR3        ; Y1 = MR3
    MSTOP                   ; end of task
```

**See also**

MMOV32 MRa, mem32 {,CNDF}



Copyright © 2024 Texas Instruments Incorporated

**MMOVF32 MRa, #32F**

***Load the 32-Bits of a 32-Bit Floating-Point Register***

## Operands

This instruction is an alias for the MMOVIZ and MMOVXI instructions. The second operand is translated by the assembler such that the instruction becomes:

```
MMOVIZ MRa, #16FHiHex MMOVXI MRa, #16FLoHex
```

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| #32F | Immediate float value represented in floating-point representation |

## Opcode

```
LSW: IIII IIII IIII IIII (opcode of MMOVIZ MRa, #16FHiHex)
MSW: 0111 1000 0100 00aa
LSW: IIII IIII IIII IIII (opcode of MMOVXI MRa, #16FLoHex)
MSW: 0111 1000 1000 00aa
```

## Description

This instruction accepts the immediate operand only in floating-point representation. To specify the immediate value as a hex value (IEEE 32-bit floating- point format), use the MOVI32 MRa, #32FHex instruction.

Load the 32-bits of MRa with the immediate float value represented by #32F.

#32F is a float value represented in floating-point representation. The assembler only accepts a float value represented in floating-point representation. That is, 3.0 can only be represented as #3.0 (#0x40400000 results in an error).

```
MRa = #32F;
```

## Flags

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

## Pipeline

Depending on #32F, this instruction takes one or two cycles. If all of the lower 16-bits of the IEEE 32-bit floating-point format of #32F are zeros, then the assembler converts MMOVF32 into only an MMOVIZ instruction. If the lower 16-bits of the IEEE 32-bit floating-point format of #32F are not zeros, then the assembler converts MMOVF32 into MMOVIZ and MMOVXI instructions.

## Example

```
MMOVF32 MR1, #3.0     ; MR1 = 3.0 (0x40400000)
                      ; Assembler converts this instruction as
                      ; MMOVIZ MR1, #0x4040
MMOVF32 MR2, #0.0     ; MR2 = 0.0 (0x00000000)
                      ; Assembler converts this instruction as
                      ; MMOVIZ MR2, #0x0
MMOVF32 MR3, #12.265  ; MR3 = 12.625 (0x41443D71)
                      ; Assembler converts this instruction as
                      ; MMOVIZ MR3, #0x4144
                      ; MMOVXI MR3, #0x3D71
```



Copyright © 2024 Texas Instruments Incorporated

**MMOVF32 MRa, #32F** (continued)

***Load the 32-Bits of a 32-Bit Floating-Point Register***

**See also**

MMOVIZ MRa, #16FHi
MMOVXI MRa, #16FLoHex
MMOVI32 MRa, #32FHex



Copyright © 2024 Texas Instruments Incorporated

**MMOVI16 MARx, #16I**

***Load the Auxiliary Register with the 16-Bit Immediate Value***

**Operands**

| | |
|---|---|
| MARx | Auxiliary register MAR0 or MAR1 |
| #16I | 16-bit immediate value |

**Opcode**

```
LSW: IIII IIII IIII IIII (opcode of MMOVI16 MAR0, #16I)
MSW: 0111 1111 1100 0000
LSW: IIII IIII IIII IIII (opcode of MMOVI16 MAR1, #16I)
MSW: 0111 1111 1110 0000
```

**Description**

Load the auxiliary register, MAR0 or MAR1, with a 16-bit immediate value. Refer to the Pipeline section for important information regarding this instruction.

```
MARx = #16I;
```

**Flags**

This instruction does not modify flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction. The immediate load of MAR0 or MAR1 occurs in the EXE phase of the pipeline. Any post increment of MAR0 or MAR1 using indirect addressing occurs in the D2 phase of the pipeline. Therefore, the following applies when loading the auxiliary registers:

- **I1 and I2**

  The two instructions following MMOVI16 use MAR0 or MAR1 before the update occurs. Thus, these two instructions use the old value of MAR0 or MAR1.
- **I3**

  Loading of an auxiliary register occurs in the EXE phase while updates due to post-increment addressing occur in the D2 phase. Thus, I3 cannot use the auxiliary register or there is a conflict. In the case of a conflict, the update due to address-mode post increment, the auxiliary register is not updated with #_X.
- **I4**

  Starting with the 4th instruction, MAR0 or MAR1 is the new value loaded with MMOVI16.

```
;  Assume MAR0 is 50 and #_X is 20
MMOVI16 MAR0, #_X             ; Load MAR0 with address of X (20)
<Instruction 1>          ; I1 Uses the old value of MAR0 (50)
<Instruction 2>          ; I2 Uses the old value of MAR0 (50)
<Instruction 3>          ; I3 Cannot use MAR0
<Instruction 4>          ; I4 Uses the new value of MAR0 (20)
<Instruction 5>          ; I5
....
```



Copyright © 2024 Texas Instruments Incorporated

**MMOVI16 MARx, #16I** (continued)

***Load the Auxiliary Register with the 16-Bit Immediate Value***

**Table 6-17. Pipeline Activity for MMOVI16 MAR0/MAR1, #16I**

| Instruction | F1 | F2 | D1 | D2 | R1 | R2 | E | W |
|---|---|---|---|---|---|---|---|---|
| MMOVI16 MAR0, #_X | MMOVI16 | | | | | | | |
| I1 | I1 | MMOVI16 | | | | | | |
| I2 | I2 | I1 | MMOVI16 | | | | | |
| I3 | I3 | I2 | I1 | MMOVI16 | | | | |
| I4 | I4 | I3 | I2 | I1 | MMOVI16 | | | |
| I5 | I5 | I4 | I3 | I2 | I1 | MMOVI16 | | |
| I6 | I6 | I5 | I4 | I3 | I2 | I1 | MMOVI16 | |



Copyright © 2024 Texas Instruments Incorporated

**MMOVI32 MRa, #32FHex**

***Load the 32-Bits of a 32-Bit Floating-Point Register with the Immediate***

**Operands**

| | |
|---|---|
| MRa | Floating-point register (MR0 to MR3) |
| #32FHex | A 32-bit immediate value that represents an IEEE 32-bit floating-point value. |

This instruction is an alias for the MMOVIZ and MMOVXI instructions. The second operand is translated by the assembler such that the instruction becomes:

```
MMOVIZ MRa, #16FHiHex
MMOVXI MRa, #16FLoHex
```

**Opcode**

```
LSW: IIII IIII IIII IIII (opcode of MMOVIZ MRa, #16FHiHex)
MSW: 0111 1000 0100 00aa
LSW: IIII IIII IIII IIII (opcode of MMOVXI MRa, #16FLoHex)
MSW: 0111 1000 1000 00aa
```

**Description**

This instruction only accepts a hex value as the immediate operand. To specify the immediate value with a floating-point representation, use the MMOVF32 MRa, #32F instruction.

Load the 32-bits of MRa with the immediate 32-bit hex value represented by #32FHex.

#32FHex is a 32-bit immediate hex value that represents the IEEE 32-bit floating-point value of a floating-point number. The assembler only accepts a hex immediate value. That is, 3.0 can only be represented as #0x40400000 (#3.0 results in an error).

```
MRa = #32FHex;
```

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

Depending on #32FHex, this instruction takes one or two cycles. If all of the lower 16-bits of #32FHex are zeros, then the assembler converts MOVI32 to an MMOVIZ instruction. If the lower 16-bits of #32FHex are not zeros, then the assembler converts MOVI32 to MMOVIZ and MMOVXI instructions.



Copyright © 2024 Texas Instruments Incorporated

**MMOVI32 MRa, #32FHex** (continued)

***Load the 32-Bits of a 32-Bit Floating-Point Register with the Immediate***

**Example**

```
MOVI32   MR1, #0x40400000 ; MR1 = 0x40400000
                          ; Assembler converts this instruction as
                          ; MMOVIZ MR1, #0x4040
MOVI32  MR2, #0x00000000  ; MR2 = 0x00000000
                          ; Assembler converts this instruction as
                          ; MMOVIZ MR2, #0x0
MOVI32  MR3, #0x40004001  ; MR3 = 0x40004001
                          ; Assembler converts this instruction as
                          ; MMOVIZ MR3, #0x4000
                          ; MMOVXI MR3, #0x4001
MOVI32  MR0, #0x00004040  ; MR0 = 0x00004040
                          ; Assembler converts this instruction as
                          ; MMOVIZ MR0, #0x0000
                          ; MMOVXI MR0, #0x4040
```

**See also**

MMOVIZ MRa, #16FHi
MMOVXI MRa, #16FLoHex
MMOVF32 MRa, #32F



Copyright © 2024 Texas Instruments Incorporated

**MMOVIZ MRa, #16FHi**

***Load the Upper 16-Bits of a 32-Bit Floating-Point Register***

**Operands**

| | |
|---|---|
| MRa | Floating-point register (MR0 to MR3) |
| #16FHi | A 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0. |

**Opcode**

```
LSW: IIII IIII IIII IIII
MSW: 0111 1000 0100 00aa
```

**Description**

Load the upper 16-bits of MRa with the immediate value #16FHi and clear the low 16-bits of MRa.

#16FHi is a 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0. The assembler only accepts a decimal or hex immediate value. That is, -1.5 can be represented as #-1.5 or #0xBFC0.

MMOVIZ is useful for loading a floating-point register with a constant in which the lowest 16-bits of the mantissa are 0. Some examples are 2.0 (0x40000000), 4.0 (0x40800000), 0.5 (0x3F000000), and -1.5 (0xBFC00000). If a constant requires all 32-bits of a floating-point register to be initialized, then use MMOVIZ along with the MMOVXI instruction.

```
MRa(31:16) = #16FHi;
MRa(15:0)  = 0;
```

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Load MR0 and MR1 with -1.5 (0xBFC00000)
     MMOVIZ    MR0, #0xBFC0      ; MR0 = 0xBFC00000 (1.5)
     MMOVIZ    MR1, #-1.5        ; MR1 = -1.5 (0xBFC00000)
; Load MR2 with pi = 3.141593 (0x40490FDB)
     MMOVIZ    MR2, #0x4049      ; MR2 = 0x40490000
     MMOVXI    MR2, #0x0FDB      ; MR2 = 0x40490FDB
```

**See also**

MMOVF32 MRa, #32F
MMOVI32 MRa, #32FHex
MMOVXI MRa, #16FLoHex



Copyright © 2024 Texas Instruments Incorporated

**MMOVZ16 MRa, mem16**

***Load MRx with 16-Bit Value***

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| mem16 | 16-bit source memory location |

**Opcode**

```
LSW: mmmm mmmm mmmm mmmm
MSW: 0111 0101 10aa addr
```

**Description**

Move the 16-bit value referenced by mem16 to the floating-point register indicated by MRa.

```
MRa(31:16) = 0;
MRa(15:0) = [mem16];
```

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | No | No |

The MSTF register flags are modified based on the integer results of the operation.

```
NF = 0;
if (MRa(31:0)== 0) { ZF = 1; }
```

**Pipeline**

This is a single-cycle instruction.



Copyright © 2024 Texas Instruments Incorporated

**MMOVXI MRa, #16FLoHex**

***Move Immediate Value to the Lower 16-Bits of a Floating-Point Register***

**Operands**

| | |
|---|---|
| MRa | CLA floating-point register (MR0 to MR3) |
| #16FLoHex | A 16-bit immediate hex value that represents the lower 16-bits of an IEEE 32-bit floating-point value. The upper 16-bits are not modified. |

**Opcode**

```
LSW: IIII IIII IIII IIII
MSW: 0111 1000 1000 00aa
```

**Description**

Load the lower 16-bits of MRa with the immediate value #16FLoHex. #16FLoHex represents the lower 16-bits of an IEEE 32-bit floating-point value. The upper 16-bits of MRa are not modified. MMOVXI can be combined with the MMOVIZ instruction to initialize all 32-bits of a MRa register.

```
MRa(15:0) = #16FLoHex;
MRa(31:16) = Unchanged;
```

**Flags**

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Load MR0 with pi = 3.141593 (0x40490FDB)
    MMOVIZ      MR0,#0x4049   ; MR0 = 0x40490000
    MMOVXI      MR0,#0x0FDB   ; MR0 = 0x40490FDB
```

**See also**

MMOVIZ MRa, #16FHi



Copyright © 2024 Texas Instruments Incorporated

## MMPYF32 MRa, MRb, MRc

### *32-Bit Floating-Point Multiply*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |
| MRc | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 00cc bbaa
MSW: 0111 1100 0000 0000
```

**Description**

Multiply the contents of two floating-point registers.

```
MRa = MRb * MRc;
```

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | Yes | Yes |

The MSTF register flags are modified as follows:

- LUF = 1 if MMPYF32 generates an underflow condition.
- LVF = 1 if MMPYF32 generates an overflow condition.

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Calculate Num/Den using a Newton-Raphson algorithum for 1/Den
; Ye = Estimate(1/X)
; Ye = Ye*(2.0 - Ye*X)
; Ye = Ye*(2.0 - Ye*X)
;
_Cla1Task1:
    MMOV32     MR1, @_Den      ; MR1 = Den
    MEINVF32   MR2, MR1        ; MR2 = Ye = Estimate(1/Den)
    MMPYF32    MR3, MR2, MR1   ; MR3 = Ye*Den
    MSUBF32    MR3, #2.0, MR3  ; MR3 = 2.0 - Ye*Den
    MMPYF32    MR2, MR2, MR3   ; MR2 = Ye = Ye*(2.0 - Ye*Den)
    MMPYF32    MR3, MR2, MR1   ; MR3 = Ye*Den
||  MMOV32     MR0, @_Num      ; MR0 = Num
    MSUBF32    MR3, #2.0, MR3  ; MR3 = 2.0 - Ye*Den
    MMPYF32    MR2, MR2, MR3   ; MR2 = Ye = Ye*(2.0 - Ye*Den)
||  MMOV32     MR1, @_Den      ; Reload Den To Set Sign
    MNEGF32    MR0, MR0, EQ    ; if(Den == 0.0) Change Sign Of Num
    MMPYF32    MR0, MR2, MR0   ; MR0 = Y = Ye*Num
    MMOV32     @_Dest, MR0     ; Store result
    MSTOP                      ; end of task
```

**See also**

MMPYF32 MRa, #16FHi, MRb
MMPYF32 MRa, MRb, MRc || MADDF32 MRd, MRe, MRf
MMPYF32 MRd, MRe, MRf || MMOV32 MRa, mem32
MMPYF32 MRd, MRe, MRf || MMOV32 mem32, MRa
MMPYF32 MRa, MRb, MRc || MSUBF32 MRd, MRe, MRf
MMACF32 MR3, MR2, MRd, MRe, MRf || MMOV32 MRa, mem32



Copyright © 2024 Texas Instruments Incorporated

**MMPYF32 MRa, #16FHi, MRb**

***32-Bit Floating-Point Multiply***

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| #16FHi | A 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The lower 16-bits of the mantissa are assumed to be all 0. |
| MRb | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: IIII IIII IIII IIII
MSW: 0111 0111 1000 bbaa
```

**Description**

Multiply MRb with the floating-point value represented by the immediate operand. Store the result of the addition in MRa.

#16FHi is a 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The lower 16-bits of the mantissa are assumed to be all 0. #16FHi is most useful for representing constants where the lowest 16-bits of the mantissa are 0. Some examples are 2.0 (0x40000000), 4.0 (0x40800000), 0.5 (0x3F000000), and -1.5 (0xBFC00000). The assembler accepts either a hex or float as the immediate value. That is, the value -1.5 can be represented as #-1.5 or #0xBFC0.

```
MRa = MRb * #16FHi:0;
```

This instruction can also be written as MMPYF32 MRa, MRb, #16FHi.

**Flags**

This instruction modifies the following flags in the MSTF register:.

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | Yes | Yes |

The MSTF register flags are modified as follows:

- LUF = 1 if MMPYF32 generates an underflow condition.
- LVF = 1 if MMPYF32 generates an overflow condition.

**Pipeline**

This is a single-cycle instruction.

**Example 1**

```
; Same as example 2 but #16FHi is represented in float
    MMOVIZ    MR3, #2.0       ; MR3 = 2.0 (0x40000000)
    MMPYF32   MR0, #3.0, MR3 ; MR0 = 3.0 * MR3 = 6.0 (0x40C00000)
    MMOV32    @_X, MR0        ; Save the result in variable X
```

**Example 2**

```
; Same as example 1 but #16FHi is represented in Hex
    MMOVIZ    MR3, #2.0          ; MR3 = 2.0 (0x40000000)
    MMPYF32   MR0, #0x4040, MR3 ; MR0 = 0x4040 * MR3 = 6.0 (0x40C00000)
    MMOV32    @_X, MR0           ; Save the result in variable X
```



Copyright © 2024 Texas Instruments Incorporated

**MMPYF32 MRa, #16FHi, MRb** (continued)

### *32-Bit Floating-Point Multiply*

**Example 3**

```
; Given X, M, and B are IQ24 numbers:
; X = IQ24(+2.5) = 0x02800000
; M = IQ24(+1.5) = 0x01800000
; B = IQ24(-0.5) = 0xFF800000
;
; Calculate Y = X * M + B
;
;
_Cla1Task2:
;
; Convert M, X, and B from IQ24 to float
     MI32TOF32   MR0, @_M          ; MR0 = 0x4BC00000
     MI32TOF32   MR1, @_X          ; MR1 = 0x4C200000
     MI32TOF32   MR2, @_B          ; MR2 = 0xCB000000
     MMPYF32     MR0, MR0, #0x3380 ; M = 1/(1*2^24) * iqm = 1.5 (0x3FC00000)
     MMPYF32     MR1, MR1, #0x3380 ; X = 1/(1*2^24) * iqx = 2.5 (0x40200000)
     MMPYF32     MR2, MR2, #0x3380 ; B = 1/(1*2^24) * iqb = -.5 (0xBF000000)
     MMPYF32     MR3, MR0, MR1     ; M*X
     MADDF32     MR2, MR2, MR3     ; Y=MX+B = 3.25 (0x40500000)
; Convert Y from float32 to IQ24
     MMPYF32     MR2, MR2, #0x4B80 ; Y * 1*2^24
     MF32TOI32   MR2, MR2          ; IQ24(Y) = 0x03400000
     MMOV32 @_Y, MR2               ; store result
     MSTOP                         ; end of task
```

**See also**

MMPYF32 MRa, MRb, #16FHi
MMPYF32 MRa, MRb, MRc
MMPYF32 MRa, MRb, MRc || MADDF32 MRd, MRe, MRf



Copyright © 2024 Texas Instruments Incorporated

**MMPYF32 MRa, MRb, #16FHi**

***32-Bit Floating-Point Multiply***

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |
| #16FHi | A 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The lower 16-bits of the mantissa are assumed to be all 0. |

**Opcode**

```
LSW: IIII IIII IIII IIII
MSW: 0111 0111 1000 bbaa
```

**Description**

Multiply MRb with the floating-point value represented by the immediate operand. Store the result of the addition in MRa.

#16FHi is a 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0. #16FHi is most useful for representing constants where the lowest 16-bits of the mantissa are 0. Some examples are 2.0 (0x40000000), 4.0 (0x40800000), 0.5 (0x3F000000), and -1.5 (0xBFC00000). The assembler accepts either a hex or float as the immediate value. That is, the value -1.5 can be represented as #-1.5 or #0xBFC0.

```
MRa = MRb * #16FHi:0;
```

This instruction can also be written as MMPYF32 MRa, #16FHi, MRb.

**Flags**

This instruction modifies the following flags in the MSTF register:.

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | Yes | Yes |

The MSTF register flags are modified as follows:

- LUF = 1 if MMPYF32 generates an underflow condition.
- LVF = 1 if MMPYF32 generates an overflow condition.

**Pipeline**

This is a single-cycle instruction.

**Example 1**

```
;Same as example 2 but #16FHi is represented in float
    MMOVIZ    MR3, #2.0        ; MR3 = 2.0 (0x40000000)
    MMPYF32   MR0, MR3, #3.0   ; MR0 = MR3 * 3.0 = 6.0 (0x40C00000)
    MMOV32    @_X, MR0         ; Save the result in variable X
```

**Example 2**

```
;Same as example 1 but #16FHi is represented in Hex
    MMOVIZ    MR3, #2.0          ; MR3 = 2.0 (0x40000000)
    MMPYF32   MR0, MR3, #0x4040  ; MR0 = MR3 * 0x4040 = 6.0 (0x40C00000)
    MMOV32    @_X, MR0           ; Save the result in variable X
```



Copyright © 2024 Texas Instruments Incorporated

**MMPYF32 MRa, MRb, #16FHi** (continued)

### *32-Bit Floating-Point Multiply*

**Example 3**

```
; Given X, M, and B are IQ24 numbers:
; X = IQ24(+2.5) = 0x02800000
; M = IQ24(+1.5) = 0x01800000
; B = IQ24(-0.5) = 0xFF800000
;
; Calculate Y = X * M + B
;
_Cla1Task2:
;
; Convert M, X, and B from IQ24 to float
    MI32TOF32   MR0, @_M          ; MR0 = 0x4BC00000
    MI32TOF32   MR1, @_X          ; MR1 = 0x4C200000
    MI32TOF32   MR2, @_B          ; MR2 = 0xCB000000
    MMPYF32     MR0, #0x3380, MR0 ; M = 1/(1*2^24) * iqm = 1.5 (0x3FC00000)
    MMPYF32     MR1, #0x3380, MR1 ; X = 1/(1*2^24) * iqx = 2.5 (0x40200000)
    MMPYF32     MR2, #0x3380, MR2 ; B = 1/(1*2^24) * iqb = -.5 (0xBF000000)
    MMPYF32     MR3, MR0, MR1     ; M*X
    MADDF32     MR2, MR2, MR3     ; Y=MX+B = 3.25 (0x40500000)
; Convert Y from float32 to IQ24
    MMPYF32     MR2, #0x4B80, MR2 ; Y * 1*2^24
    MF32TOI32   MR2, MR2          ; IQ24(Y) = 0x03400000
    MMOV32      @_Y, MR2          ; store result
    MSTOP                         ; end of task
```

**See also**

MMPYF32 MRa, #16FHi, MRb
MMPYF32 MRa, MRb, MRc



Copyright © 2024 Texas Instruments Incorporated

**MMPYF32 MRa, MRb, MRc||MADDF32 MRd, MRe, MRf**

***32-Bit Floating-Point Multiply with Parallel Add***

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register for MMPYF32 (MR0 to MR3)<br>MRa cannot be the same register as MRd |
| MRb | CLA floating-point source register for MMPYF32 (MR0 to MR3) |
| MRc | CLA floating-point source register for MMPYF32 (MR0 to MR3) |
| MRd | CLA floating-point destination register for MADDF32 (MR0 to MR3)<br>MRd cannot be the same register as MRa |
| MRe | CLA floating-point source register for MADDF32 (MR0 to MR3) |
| MRf | CLA floating-point source register for MADDF32 (MR0 to MR3) |

**Opcode**

```
LSW: 0000 ffee ddcc bbaa
MSW: 0111 1010 0000 0000
```

**Description**

Multiply the contents of two floating-point registers with parallel addition of two registers.

```
MRa = MRb * MRc;
MRd = MRe + MRf;
```

**Restrictions**

The destination register for the MMPYF32 and the MADDF32 must be unique. That is, MRa cannot be the same register as MRd.

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | Yes | Yes |

The MSTF register flags are modified as follows:

- LUF = 1 if MMPYF32 or MADDF32 generates an underflow condition.
- LVF = 1 if MMPYF32 or MADDF32 generates an overflow condition.

**Pipeline**

Both MMPYF32 and MADDF32 complete in a single cycle.



Copyright © 2024 Texas Instruments Incorporated

**MMPYF32 MRa, MRb, MRc||MADDF32 MRd, MRe, MRf** (continued)

### *32-Bit Floating-Point Multiply with Parallel Add*

**Example**

```
; Perform 5 multiply and accumulate operations:
;
; X and Y are 32-bit floating-point arrays
;
; 1st multiply: A = X0 * Y0
; 2nd multiply: B = X1 * Y1
; 3rd multiply: C = X2 * Y2
; 4th multiply: D = X3 * Y3
; 5th multiply: E = X3 * Y3
;
; Result = A + B + C + D + E
;
_Cla1Task1:
    MMOVI16    MAR0, #_X               ; MAR0 points to X array
    MMOVI16    MAR1, #_Y               ; MAR1 points to Y array
    MNOP                               ; Delay for MAR0, MAR1 load
    MNOP                               ; Delay for MAR0, MAR1 load
                                       ; <-- MAR0 valid
    MMOV32     MR0, *MAR0[2]++         ; MR0 = X0, MAR0 += 2
                                       ; <-- MAR1 valid
    MMOV32     MR1, *MAR1[2]++         ; MR1 = Y0, MAR1 += 2
    MMPYF32    MR2, MR0, MR1           ; MR2 = A = X0 * Y0
||  MMOV32     MR0, *MAR0[2]++         ; In parallel MR0 = X1, MAR0 += 2
    MMOV32     MR1, *MAR1[2]++         ; MR1 = Y1, MAR1 += 2
    MMPYF32    MR3, MR0, MR1           ; MR3 = B = X1 * Y1
||  MMOV32     MR0, *MAR0[2]++         ; In parallel MR0 = X2, MAR0 += 2
    MMOV32     MR1, *MAR1[2]++         ; MR1 = Y2, MAR2 += 2
    MMACF32    MR3, MR2, MR2, MR0, MR1 ; MR3 = A + B, MR2 = C = X2 * Y2
||  MMOV32     MR0, *MAR0[2]++         ; In parallel MR0 = X3
    MMOV32     MR1, *MAR1[2]++         ; MR1 = Y3
    MMACF32    MR3, MR2, MR2, MR0, MR1 ; MR3 = (A + B) + C, MR2 = D = X3 * Y3
||  MMOV32     MR0, *MAR0              ; In parallel MR0 = X4
    MMOV32     MR1, *MAR1              ; MR1 = Y4
    MMPYF32    MR2, MR0, MR1           ; MR2 = E = X4 * Y4
||  MADDF32    MR3, MR3, MR2           ; in parallel MR3 = (A + B + C) + D

    MADDF32    MR3, MR3, MR2           ; MR3 = (A + B + C + D) + E
    MMOV32     @_Result, MR3           ; Store the result
    MSTOP                              ; end of task
```

**See also**

MMACF32 MR3, MR2, MRd, MRe, MRf || MMOV32 MRa, mem32



Copyright © 2024 Texas Instruments Incorporated

**MMPYF32 MRd, MRe, MRf ||MMOV32 MRa, mem32**

***32-Bit Floating-Point Multiply with Parallel Move***

**Operands**

| | |
|---|---|
| MRd | CLA floating-point destination register for MMPYF32 (MR0 to MR3)<br>MRd cannot be the same register as MRa |
| MRe | CLA floating-point source register for MMPYF32 (MR0 to MR3) |
| MRf | CLA floating-point source register for MMPYF32 (MR0 to MR3) |
| MRa | CLA floating-point destination register for MMOV32 (MR0 to MR3)<br>MRa cannot be the same register as MRd |
| mem32 | 32-bit memory location accessed using one of the available addressing modes. This is the source of MMOV32. |

**Opcode**

```
LSW: mmmm mmmm mmmm mmmm
MSW: 0000 ffee ddaa addr
```

**Description**

Multiply the contents of two floating-point registers and load another.

```
MRd = MRe * MRf;
MRa = [mem32];
```

**Restrictions**

The destination register for the MMPYF32 and the MMOV32 must be unique. That is, MRa cannot be the same register as MRd.

**Flags**

This instruction modifies the following flags in the MSTF register:.

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | Yes | Yes |

The MSTF register flags are modified as follows:

- LUF = 1 if MMPYF32 generates an underflow condition.
- LVF = 1 if MMPYF32 generates an overflow condition.

The MMOV32 instruction sets the NF and ZF flags as follows:

```
NF = MRa(31);
ZF = 0;
if(MRa(30:23) == 0) { ZF = 1; NF = 0; }
```

**Pipeline**

Both MMPYF32 and MMOV32 complete in a single cycle.

**Example 1**

```
; Given M1, X1, and B1 are 32-bit floating point
; Calculate Y1 = M1*X1+B1
;
_Cla1Task1:
    MMOV32     MR0, @M1        ; Load MR0 with M1
    MMOV32     MR1, @X1        ; Load MR1 with X1
    MMPYF32    MR1, MR1, MR0   ; Multiply M1*X1
||  MMOV32     MR0, @B1        ; and in parallel load MR0 with B1
    MADDF32    MR1, MR1, MR0   ; Add M*X1 to B1 and store in MR1
    MMOV32     @Y1, MR1        ; Store the result
    MSTOP                      ; end of task
```



Copyright © 2024 Texas Instruments Incorporated

**MMPYF32 MRd, MRe, MRf ||MMOV32 MRa, mem32** (continued)

### *32-Bit Floating-Point Multiply with Parallel Move*

**Example 2**

```
; Given A, B, and C are 32-bit floating-point numbers
; Calculate Y2 = (A * B)
;           Y3 = (A * B) * C
;
_Cla1Task2:
    MMOV32    MR0, @A         ; Load MR0 with A
    MMOV32    MR1, @B         ; Load MR1 with B
    MMPYF32   MR1, MR1, MR0   ; Multiply A*B
||  MMOV32    MR0, @C         ; and in parallel load MR0 with C
    MMPYF32   MR1, MR1, MR0   ; Multiply (A*B) by C
||  MMOV32    @Y2, MR1        ; and in parallel store A*B
    MMOV32    @Y3, MR1        ; Store the result
    MSTOP                     ; end of task
```

**See also**

MMPYF32 MRd, MRe, MRf || MMOV32 mem32, MRa
MMACF32 MR3, MR2, MRd, MRe, MRf || MMOV32 MRa, mem32



Copyright © 2024 Texas Instruments Incorporated

## MMPYF32 MRd, MRe, MRf ||MMOV32 mem32, MRa

### *32-Bit Floating-Point Multiply with Parallel Move*

**Operands**

| | |
|---|---|
| MRd | CLA floating-point destination register for MMPYF32 (MR0 to MR3) |
| MRe | CLA floating-point source register for MMPYF32 (MR0 to MR3) |
| MRf | CLA floating-point source register for MMPYF32 (MR0 to MR3) |
| mem32 | 32-bit memory location accessed using one of the available addressing modes. This is the destination of MMOV32. |
| MRa | CLA floating-point source register for MMOV32 (MR0 to MR3) |

**Opcode**

```
LSW: mmmm mmmm mmmm mmmm
MSW: 0100 ffee ddaa addr
```

**Description**

Multiply the contents of two floating-point registers and move from memory to register.

```
MRd = MRe * MRf;
[mem32] = MRa;
```

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | Yes | Yes |

The MSTF register flags are modified as follows:

- LUF = 1 if MMPYF32 generates an underflow condition.
- LVF = 1 if MMPYF32 generates an overflow condition.

**Pipeline**

MMPYF32 and MMOV32 both complete in a single cycle.

**Example**

```
; Given A, B, and C are 32-bit floating-point numbers
; Calculate Y2 = (A * B)
;           Y3 = (A * B) * C
;
_Cla1Task2:
     MMOV32    MR0, @A         ; Load MR0 with A
     MMOV32    MR1, @B         ; Load MR1 with B
     MMPYF32   MR1, MR1, MR0   ; Multiply A*B
||   MMOV32    MR0, @C         ; and in parallel load MR0 with C
     MMPYF32   MR1, MR1, MR0   ; Multiply (A*B) by C
||   MMOV32    @Y2, MR1        ; and in parallel store A*B
     MMOV32    @Y3, MR1        ; Store the result
     MSTOP                     ; end of task
```

**See also**

MMPYF32 MRd, MRe, MRf || MMOV32 MRa, mem32
MMACF32 MR3, MR2, MRd, MRe, MRf || MMOV32 MRa, mem32



Copyright © 2024 Texas Instruments Incorporated

**MMPYF32 MRa, MRb, MRc ||MSUBF32 MRd, MRe, MRf**

***32-Bit Floating-Point Multiply with Parallel Subtract***

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register for MMPYF32 (MR0 to MR3)<br>MRa cannot be the same register as MRd |
| MRb | CLA floating-point source register for MMPYF32 (MR0 to MR3) |
| MRc | CLA floating-point source register for MMPYF32 (MR0 to MR3) |
| MRd | CLA floating-point destination register for MSUBF32 (MR0 to MR3)<br>MRd cannot be the same register as MRa |
| MRe | CLA floating-point source register for MSUBF32 (MR0 to MR3) |
| MRf | CLA floating-point source register for MSUBF32 (MR0 to MR3) |

**Opcode**

```
LSW: 0000 ffee ddcc bbaa
MSW: 0111 1010 0100 0000
```

**Description**

Multiply the contents of two floating-point registers with parallel subtraction of two registers.

```
MRa = MRb * MRc;
MRd = MRe - MRf;
```

**Restrictions**

The destination register for the MMPYF32 and the MSUBF32 must be unique. That is, MRa cannot be the same register as MRd.

**Flags**

This instruction modifies the following flags in the MSTF register:.

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | Yes | Yes |

The MSTF register flags are modified as follows:

- LUF = 1 if MMPYF32 or MSUBF32 generates an underflow condition.
- LVF = 1 if MMPYF32 or MSUBF32 generates an overflow condition.

**Pipeline**

MMPYF32 and MSUBF32 both complete in a single cycle.

**Example**

```
; Given A, B, and C are 32-bit floating-point numbers
; Calculate Y2 = (A * B)
;           Y3 = (A - B)
;
_Cla1Task2:
     MMOV32   MR0, @A          ; Load MR0 with A
     MMOV32   MR1, @B          ; Load MR1 with B
     MMPYF32  MR2, MR0, MR1    ; Multiply (A*B)
||   MSUBF32  MR3, MR0, MR1    ; and in parallel Sub (A-B)
     MMOV32   @Y2, MR2         ; Store A*B
     MMOV32   @Y3, MR3         ; Store A-B
     MSTOP                     ; end of task
```



Copyright © 2024 Texas Instruments Incorporated

**MMPYF32 MRa, MRb, MRc ||MSUBF32 MRd, MRe, MRf** (continued)

***32-Bit Floating-Point Multiply with Parallel Subtract***

**See also**

MSUBF32 MRa, MRb, MRc
MSUBF32 MRd, MRe, MRf || MMOV32 MRa, mem32
MSUBF32 MRd, MRe, MRf || MMOV32 mem32, MRa



Copyright © 2024 Texas Instruments Incorporated

**MNEGF32 MRa, MRb{, CNDF}**

***Conditional Negation***

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |
| CNDF | Condition tested |

**Opcode**

```
LSW: 0000 0000 cndf bbaa
MSW: 0111 1010 1000 0000
```

**Description**

```
if (CNDF == true) {MRa = - MRb; }
else {MRa = MRb; }
```

CNDF is one of the following conditions:

| Encode[1] | CNDF | Description | MSTF Flags Tested |
|---|---|---|---|
| 0000 | NEQ | Not equal to zero | ZF == 0 |
| 0001 | EQ | Equal to zero | ZF == 1 |
| 0010 | GT | Greater than zero | ZF == 0 AND NF == 0 |
| 0011 | GEQ | Greater than or equal to zero | NF == 0 |
| 0100 | LT | Less than zero | NF == 1 |
| 0101 | LEQ | Less than or equal to zero | ZF == 1 OR NF == 1 |
| 1010 | TF | Test flag set | TF == 1 |
| 1011 | NTF | Test flag not set | TF == 0 |
| 1100 | LU | Latched underflow | LUF == 1 |
| 1101 | LV | Latched overflow | LVF == 1 |
| 1110 | UNC | Unconditional | None |
| 1111 | UNCF[2] | Unconditional with flag modification | None |

(1) Values not shown are reserved.
(2) This is the default operation, if no CNDF field is specified. This condition allows the ZF, and NF flags to be modified when a conditional operation is executed. All other conditions do not modify these flags.

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | No | No |

**Pipeline**

This is a single-cycle instruction.



Copyright © 2024 Texas Instruments Incorporated

**MNEGF32 MRa, MRb{, CNDF}** (continued)

***Conditional Negation***

**Example 1**

```
; Show the basic operation of MNEGF32
;
     MMOVIZ    MR0, #5.0       ; MR0 = 5.0 (0x40A00000)
     MMOVIZ    MR1, #4.0       ; MR1 = 4.0 (0x40800000)
     MMOVIZ    MR2, #-1.5      ; MR2 = -1.5 (0xBFC00000)
     MMPYF32   MR3, MR1, MR2   ; MR3 = -6.0
     MMPYF32   MR0, MR0, MR1   ; MR0 = 20.0
     MMOVIZ    MR1, #0.0
     MCMPF32   MR3, MR1        ; NF = 1
     MNEGF32   MR3, MR3, LT    ; if NF = 1, MR3 = 6.0
     MCMPF32   MR0, MR1        ; NF = 0
     MNEGF32   MR0, MR0, GEQ   ; if NF = 0, MR0 = -20.0
```

**Example 2**

```
; Calculate Num/Den using a Newton-Raphson algorithum for 1/Den
; Ye = Estimate(1/X)
; Ye = Ye*(2.0 - Ye*X)
; Ye = Ye*(2.0 - Ye*X)
;
_Cla1Task1:
    MMOV32    MR1, @_Den      ; MR1 = Den
    MEINVF32  MR2, MR1        ; MR2 = Ye = Estimate(1/Den)
    MMPYF32   MR3, MR2, MR1   ; MR3 = Ye*Den
    MSUBF32   MR3, #2.0, MR3  ; MR3 = 2.0 - Ye*Den
    MMPYF32   MR2, MR2, MR3   ; MR2 = Ye = Ye*(2.0 - Ye*Den)
    MMPYF32   MR3, MR2, MR1   ; MR3 = Ye*Den
||  MMOV32    MR0, @_Num      ; MR0 = Num
    MSUBF32   MR3, #2.0, MR3  ; MR3 = 2.0 - Ye*Den
    MMPYF32   MR2, MR2, MR3   ; MR2 = Ye = Ye*(2.0 - Ye*Den)
||  MMOV32    MR1, @_Den      ; Reload Den To Set Sign
    MNEGF32   MR0, MR0, EQ    ; if(Den == 0.0) Change Sign Of Num
    MMPYF32   MR0, MR2, MR0   ; MR0 = Y = Ye*Num
    MMOV32    @_Dest, MR0     ; Store result
    MSTOP                     ; end of task
```

**See also**

MABSF32 MRa, MRb



Copyright © 2024 Texas Instruments Incorporated

## MNOP

***No Operation***

### Operands

| | |
|---|---|
| none | This instruction does not have any operands |

### Opcode

```
LSW: 0000 0000 0000 0000
MSW: 0111 1111 1010 0000
```

### Description

Do nothing. This instruction is used to fill required pipeline delay slots when other instructions are not available to fill the slots.

### Flags

This instruction does not modify flags in the MSTF register.

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

### Pipeline

This is a single-cycle instruction.

### Example

```
; X is an array of 32-bit floating-point values
; Find the maximum value in an array X
; and store the value in Result
;
_Cla1Task1:
    MMOVI16    MAR1,#_X          ; Start address
    MUI16TOF32 MR0, @_len        ; Length of the array
    MNOP                         ; delay for MAR1 load
    MNOP                         ; delay for MAR1 load
    MMOV32     MR1, *MAR1[2]++   ; MR1 = X0
LOOP
    MMOV32     MR2, *MAR1[2]++   ; MR2 = next element
    MMAXF32    MR1, MR2          ; MR1 = MAX(MR1, MR2)
    MADDF32    MR0, MR0, #-1.0   ; Decrememt the counter
    MCMPF32    MR0 #0.0          ; Set/clear flags for MBCNDD
    MNOP                         ; Too late to affect MBCNDD
    MNOP                         ; Too late to affect MBCNDD
    MNOP                         ; Too late to affect MBCNDD
    MBCNDD    LOOP, NEQ          ; Branch if not equal to zero
    MMOV32    @_Result, MR1      ; Always executed
    MNOP                         ; Pad to seperate MBCNDD and MSTOP
    MNOP                         ; Pad to seperate MBCNDD and MSTOP
    MSTOP                        ; End of task
```



Copyright © 2024 Texas Instruments Incorporated

### MOR32 MRa, MRb, MRc

***Bitwise OR***

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |
| MRc | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 00cc bbaa
MSW: 0111 1100 1000 0000
```

**Description**

Bitwise OR of MRb with MRc.

```
MARa(31:0) = MARb(31:0) OR MRc(31:0);
```

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | No | No |

The MSTF register flags are modified based on the integer results of the operation.

```
NF = MRa(31);
ZF = 0;
if(MRa(31:0) == 0) { ZF = 1; }
```

**Pipeline**

This is a single-cycle instruction.

**Example**

```
MMOVIZ    MR0, #0x5555 ; MR0 = 0x5555AAAA
MMOVXI    MR0, #0xAAAA
MMOVIZ    MR1, #0x5432 ; MR1 = 0x5432FEDC
MMOVXI    MR1, #0xFEDC
                       ; 0101 OR 0101 = 0101 (5)
                       ; 0101 OR 0100 = 0101 (5)
                       ; 0101 OR 0011 = 0111 (7)
                       ; 0101 OR 0010 = 0111 (7)
                       ; 1010 OR 1111 = 1111 (F)
                       ; 1010 OR 1110 = 1110 (E)
                       ; 1010 OR 1101 = 1111 (F)
                       ; 1010 OR 1100 = 1110 (E)
MOR32 MR2, MR1, MR0    ; MR3 = 0x5555FEFE
```

**See also**

MAND32 MRa, MRb, MRc
MXOR32 MRa, MRb, MRc



Copyright © 2024 Texas Instruments Incorporated

**MRCNDD {CNDF}**

***Return Conditional Delayed***

**Operands**

| | |
|---|---|
| CNDF | Optional condition |

**Opcode**

```
LSW: 0000 0000 0000 0000
MSW: 0111 1001 1010 cndf
```

**Description**

If the specified condition is true, then the RPC field of MSTF is loaded into MPC and fetching continues from that location. Otherwise, program fetches continue without the return.

Refer to the Pipeline section for important information regarding this instruction.

```
if (CNDF == TRUE) MPC = RPC;
```

CNDF is one of the following conditions:

| Encode(1) | CNDF | Description | MSTF Flags Tested |
|---|---|---|---|
| 0000 | NEQ | Not equal to zero | ZF == 0 |
| 0001 | EQ | Equal to zero | ZF == 1 |
| 0010 | GT | Greater than zero | ZF == 0 AND NF == 0 |
| 0011 | GEQ | Greater than or equal to zero | NF == 0 |
| 0100 | LT | Less than zero | NF == 1 |
| 0101 | LEQ | Less than or equal to zero | ZF == 1 OR NF == 1 |
| 1010 | TF | Test flag set | TF == 1 |
| 1011 | NTF | Test flag not set | TF == 0 |
| 1100 | LU | Latched underflow | LUF == 1 |
| 1101 | LV | Latched overflow | LVF == 1 |
| 1110 | UNC | Unconditional | None |
| 1111 | UNCF(2) | Unconditional with flag modification | None |

(1) Values not shown are reserved.
(2) This is the default operation, if no CNDF field is specified. This condition allows the ZF and NF flags to be modified when a conditional operation is executed. All other conditions do not modify these flags.

**Flags**

This instruction does not modify flags in the MSTF register.

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

The MRCNDD instruction is a single-cycle instruction. As shown in Table 6-18, 6 instruction slots are executed for each return; 3 slots before the return instruction (d5-d7) and 3 slots after the return instruction (d8-d10). The total number of cycles for a return taken or not taken depends on the usage of these slots. That is, the number of cycles depends on how many slots are filled with a MNOP as well as which slots are filled.



Copyright © 2024 Texas Instruments Incorporated

**MRCNDD {CNDF}** (continued)

***Return Conditional Delayed***

The effective number of cycles for a return can, therefore, range from 1 to 7 cycles. The number of cycles for a return taken cannot be the same as for a return not taken.

Referring to the following code fragment and the pipeline diagrams in Table 6-18 and Table 6-19, the instructions before and after MRCNDD have the following properties:

```
;
;
<Instruction 1>     ; I1 Last instruction that can affect flags for
                    ; the MCCNDD operation
<Instruction 2>     ; I2 Cannot be stop, branch, call or return
<Instruction 3>     ; I3 Cannot be stop, branch, call or return
<Instruction 4>     ; I4 Cannot be stop, branch, call or return
MCCNDD _func, NEQ   ; Call to func if not eqal to zero
                    ; Three instructions after MCCNDD are always
                    ; executed whether the call is taken or not
<Instruction 5>     ; I5 Cannot be stop, branch, call or return
<Instruction 6>     ; I6 Cannot be stop, branch, call or return
<Instruction 7>     ; I7 Cannot be stop, branch, call or return
<Instruction 8>     ; I8 The address of this instruction is saved
                    ; in the RPC field of the MSTF register.
                    ; Upon return this value is loaded into MPC
                    ; and fetching continues from this point.
<Instruction 9>     ; I9
<Instruction 10>    ; I10
....
....
_func:
<Destination 1>     ; d1 Can be any instruction
<Destination 2>     ; d2
<Destination 3>     ; d3
<Destination 4>     ; d4 Last instruction that can affect flags for
                    ; the MRCNDD operation
<Destination 5>     ; d5 Cannot be stop, branch, call or return
<Destination 6>     ; d6 Cannot be stop, branch, call or return
<Destination 7>     ; d7 Cannot be stop, branch, call or return
MRCNDD NEQ          ; Return to <Instruction 8> if not equal to zero
                    ; Three instructions after MRCNDD are always
                    ; executed whether the return is taken or not
<Destination 8>     ; d8 Cannot be stop, branch, call or return
<Destination 9>     ; d9 Cannot be stop, branch, call or return
<Destination 10>    ; d10 Cannot be stop, branch, call or return
<Destination 11>    ; d11
<Destination 12>    ; d12
....
....
MSTOP
....
```

- **d4**
  - d4 is the last instruction that can effect the CNDF flags for the MRCNDD instruction. The CNDF flags are tested in the D2 phase of the pipeline. That is, a decision is made whether to return or not when MRCNDD is in the D2 phase.
  - There are no restrictions on the type of instruction for d4.
- **d5, d6, and d7**
  - The three instructions proceeding MRCNDD can change MSTF flags but have no effect on whether the MRCNDD instruction makes the return or not. This is because the flag modification occurs after the D2 phase of the MRCNDD instruction.
  - These instructions must not be the following: MSTOP, MDEBUGSTOP, MBCNDD, MCCNDD, or MRCNDD.
- **d8, d9, and d10**
  - The three instructions following MRCNDD are always executed irrespective of whether the return is taken or not.



Copyright © 2024 Texas Instruments Incorporated

**MRCNDD {CNDF}** (continued)

***Return Conditional Delayed***

– These instructions must not be the following: MSTOP, MDEBUGSTOP, MBCNDD, MCCNDD, or MRCNDD.

**Table 6-18. Pipeline Activity for MRCNDD, Return Not Taken**

| Instruction | F1 | F2 | D1 | D2 | R1 | R2 | E | W |
|---|---|---|---|---|---|---|---|---|
| d4 | d4 | d3 | d2 | d1 | I7 | I6 | I5 | |
| d5 | d5 | d4 | d3 | d2 | d1 | I7 | I6 | |
| d6 | d6 | d5 | d4 | d3 | d2 | d1 | i7 | |
| d7 | d7 | d6 | d5 | d4 | d3 | d2 | d1 | |
| MRCNDD | MRCNDD | d7 | d6 | d5 | d4 | d3 | d2 | |
| d8 | d8 | MRCNDD | d7 | d6 | d5 | d4 | d3 | |
| d9 | d9 | d8 | MRCNDD | d7 | d6 | d5 | d4 | |
| d10 | d10 | d9 | d8 | MRCNDD | d7 | d6 | d5 | |
| d11 | d11 | d10 | d9 | d8 | - | d7 | d6 | |
| d12 | d12 | d11 | d10 | d9 | d8 | - | d7 | |
| etc.... | .... | d12 | d11 | d10 | d9 | d8 | - | |
| .... | .... | .... | d12 | d11 | d10 | d9 | d8 | |
| .... | .... | .... | .... | d12 | d11 | d10 | d9 | |
| | | | | | d12 | d11 | d10 | |
| | | | | | | d12 | d11 | |
| | | | | | | | d12 | |

**Table 6-19. Pipeline Activity for MRCNDD, Return Taken**

| Instruction | F1 | F2 | D1 | D2 | R1 | R2 | E | W |
|---|---|---|---|---|---|---|---|---|
| d4 | d4 | d3 | d2 | d1 | I7 | I6 | I5 | |
| d5 | d5 | d4 | d3 | d2 | d1 | I7 | I6 | |
| d6 | d6 | d5 | d4 | d3 | d2 | d1 | i7 | |
| d7 | d7 | d6 | d5 | d4 | d3 | d2 | d1 | |
| MRCNDD | MRCNDD | d7 | d6 | d5 | d4 | d3 | d2 | |
| d8 | d8 | MRCNDD | d7 | d6 | d5 | d4 | d3 | |
| d9 | d9 | d8 | MRCNDD | d7 | d6 | d5 | d4 | |
| d10 | d10 | d9 | d8 | MRCNDD | d7 | d6 | d5 | |
| I8 | I8 | d10 | d9 | d8 | - | d7 | d6 | |
| I9 | I9 | I8 | d10 | d9 | d8 | - | d7 | |
| I10 | I10 | I9 | I8 | d10 | d9 | d8 | - | |
| etc.... | .... | I10 | I9 | I8 | d10 | d9 | d8 | |
| .... | .... | | I10 | I9 | I8 | d10 | d9 | |
| .... | .... | | | I10 | I9 | I8 | d10 | |
| | | | | | I10 | I9 | I8 | |
| | | | | | | I10 | I9 | |
| | | | | | | | I10 | |

**See also**

MBCNDD #16BitDest, CNDF
MCCNDD 16BitDest, CNDF
MMOV32 mem32, MSTF
MMOV32 MSTF, mem32



Copyright © 2024 Texas Instruments Incorporated

## MSETFLG FLAG, VALUE

***Set or Clear Selected Floating-Point Status Flags***

### Operands

| | |
|---|---|
| FLAG | 8-bit mask indicating which floating-point status flags to change. |
| VALUE | 8-bit mask indicating the flag value: 0 or 1. |

### Opcode

```
LSW: FFFF FFFF VVVV VVVV
MSW: 0111 1001 1100 0000
```

### Description

The MSETFLG instruction is used to set or clear selected floating-point status flags in the MSTF register. The FLAG field is an 11-bit value that indicates which flags are changed. That is, if a FLAG bit is set to 1, that flag is changed; all other flags are not modified. The bit mapping of the FLAG field is:

| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| RNDF32 | Reserved | | TF | Reserved | | ZF | NF | LUF | LVF |

The VALUE field indicates the value the flag can be set to: 0 or 1.

### Flags

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | Yes | Yes | Yes | Yes | Yes |

Any flag can be modified by this instruction. The MEALLOW and RPC fields cannot be modified with this instruction.

### Pipeline

This is a single-cycle instruction.

### Example

To make it easier and legible, the assembler accepts a FLAG=VALUE syntax for the MSTFLG operation as:

```
MSETFLG RNDF32=0, TF=0, NF=1; FLAG = 11000100; VALUE = 00XXX1XX;
```

### See also

MMOV32 mem32, MSTF
MMOV32 MSTF, mem32



Copyright © 2024 Texas Instruments Incorporated

**MSTOP**

***Stop Task***

**Operands**

| | |
|---|---|
| none | This instruction does not have any operands |

**Opcode**

```
LSW: 0000 0000 0000 0000
MSW: 0111 1111 1000 0000
```

**Description**

The MSTOP instruction must be placed to indicate the end of each task. In addition, placing MSTOP in unused memory locations within the CLA program RAM can be useful for debugging and preventing run away CLA code. When MSTOP enters the D2 phase of the pipeline, the MIRUN flag for the task is cleared and the associated interrupt is flagged in the PIE vector table.

There are three special cases that can occur when single-stepping a task such that the MPC reaches the MSTOP instruction.

1. If you are single-stepping or halted in "task A" and "task B" comes in before the MPC reaches the MSTOP, then "task B" starts if you continue to step through the MSTOP instruction. Basically, if "task B" is pending before the MPC reaches MSTOP in "task A" then there is no issue in "task B" starting and no special action is required.
2. In this case, you have single-stepped or halted in "task A" and the MPC has reached the MSTOP with no tasks pending. If "task B" comes in at this point, "task B" is flagged in the MIFR register but "task B" can or cannot start if you continue to single-step through the MSTOP instruction of "task A". It depends on exactly when the new task comes in. To reliably start "task B", perform a soft reset and reconfigure the MIER bits. Once this is done, you can start single-stepping "task B".
3. Case 2 can be handled slightly differently if there is control over when "task B" comes in (for example using the IACK instruction to start the task). In this case you have single-stepped or halted in "task A" and the MPC has reached the MSTOP with no tasks pending. Before forcing "task B", run free to force the CLA out of the debug state. Once this is done you can force "task B" and continue debugging.

**Restrictions**

The MSTOP instruction cannot be placed 3 instructions before or after a MBCNDD, MCCNDD, or MRCNDD instruction.

**Flags**

This instruction does not modify flags in the MSTF register.

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |



Copyright © 2024 Texas Instruments Incorporated

**MSTOP** (continued)

***Stop Task***

**Pipeline**

This is a single-cycle instruction. Table 6-20 shows the pipeline behavior of the MSTOP instruction. The MSTOP instruction cannot be placed with 3 instructions of a MBCNDD, MCCNDD, or MRCNDD instruction.

**Table 6-20. Pipeline Activity for MSTOP**

| Instruction | F1 | F2 | D1 | D2 | R1 | R2 | E | W |
|---|---|---|---|---|---|---|---|---|
| I1 | I1 | | | | | | | |
| I2 | I2 | I1 | | | | | | |
| I3 | I3 | I2 | I1 | | | | | |
| MSTOP | MSTOP | I3 | I2 | I1 | | | | |
| I4 | I4 | MSTOP | I3 | I2 | I1 | | | |
| I5 | I5 | I4 | MSTOP | I3 | I2 | I1 | | |
| I6 | I6 | I5 | I4 | MSTOP | I3 | I2 | I1 | |
| New Task Arbitrated and Prioritized | - | - | - | - | - | I3 | I2 | |
| New Task Arbitrated and Prioritized | - | - | - | - | - | - | I3 | |
| I1 | I1 | - | - | - | - | - | - | |
| I2 | I2 | I1 | - | - | - | - | - | |
| I3 | I3 | I2 | I1 | - | - | - | - | |
| I4 | I4 | I3 | I2 | I1 | - | - | - | |
| I5 | I5 | I4 | I3 | I2 | I1 | - | - | |
| I6 | I6 | I5 | I4 | I3 | I2 | I1 | - | |
| I7 | I7 | I6 | I5 | I4 | I3 | I2 | I1 | |
| .... | | | | | | | | |

**Example**

```
; Given A = (int32)1
;       B = (int32)2
;       C = (int32)-7
;
; Calculate Y2 = A - B - C
_Cla1Task3:
   MMOV32   MR0, @_A       ; MR0 = 1 (0x00000001)
   MMOV32   MR1, @_B       ; MR1 = 2 (0x00000002)
   MMOV32   MR2, @_C       ; MR2 = -7 (0xFFFFFFF9)
   MSUB32   MR3, MR0, MR1 ; A + B
   MSUB32   MR3, MR3, MR2 ; A + B + C = 6 (0x0000006)
   MMOV32   @_y2, MR3      ; Store y2
   MSTOP                   ; End of task
```

**See also**

MDEBUGSTOP



Copyright © 2024 Texas Instruments Incorporated

## MSUB32 MRa, MRb, MRc

### *32-Bit Integer Subtraction*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point destination register (MR0 to MR3) |
| MRc | CLA floating-point destination register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 00cc bbaa
MSW: 0111 1100 1110 0000
```

**Description**

32-bit integer addition of MRb and MRc.

```
MARa(31:0) = MARb(31:0) - MRc(31:0);
```

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | No | No |

The MSTF register flags are modified as follows:

```
NF = MRa(31);
ZF = 0;
if(MRa(31:0) == 0) { ZF = 1; }
```

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Given A = (int32)1
;       B = (int32)2
;       C = (int32)-7
;
;
Calculate Y2 = A - B - C
;
_Cla1Task3:
   MMOV32   MR0, @_A          ; MR0 = 1 (0x00000001)
   MMOV32   MR1, @_B          ; MR1 = 2 (0x00000002)
   MMOV32   MR2, @_C          ; MR2 = -7 (0xFFFFFFF9)
   MSUB32   MR3, MR0, MR1     ; A + B
   MSUB32   MR3, MR3, MR2     ; A + B + C = 6 (0x0000006)
   MMOV32   @_y2, MR3         ; Store y2
   MSTOP                      ; End of task
```

**See also**

MADD32 MRa, MRb, MRc
MAND32 MRa, MRb, MRc
MASR32 MRa, #SHIFT
MLSL32 MRa, #SHIFT
MLSR32 MRa, #SHIFT
MOR32 MRa, MRb, MRc
MXOR32 MRa, MRb, MRc



Copyright © 2024 Texas Instruments Incorporated

### MSUBF32 MRa, MRb, MRc

*32-Bit Floating-Point Subtraction*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to R1) |
| MRb | CLA floating-point source register (MR0 to R1) |
| MRc | CLA floating-point source register (MR0 to R1) |

**Opcode**

```
LSW: 0000 0000 00cc bbaa
MSW: 0111 1100 0100 0000
```

**Description**

Subtract the contents of two floating-point registers

```
MRa = MRb - MRc;
```

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | Yes | Yes |

The MSTF register flags are modified as follows:

- LUF = 1 if MSUBF32 generates an underflow condition.
- LVF = 1 if MSUBF32 generates an overflow condition.

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Given A, B, and C are 32-bit floating-point numbers
; Calculate Y2 = A + B - C
;
_Cla1Task5:
   MMOV32    MR0, @_A      ; Load MR0 with A
   MMOV32    MR1, @_B      ; Load MR1 with B
   MADDF32   MR0, MR1, MR0 ; Add A + B
|| MMOV32    MR1, @_C      ;   and in parallel load C
   MSUBF32   MR0, MR0, MR1 ; Subtract C from (A + B)
   MMOV32    @Y,  MR0      ; (A+B) - C
   MSTOP                   ; end of task
```

**See also**

MSUBF32 MRa, #16FHi, MRb
MSUBF32 MRd, MRe, MRf || MMOV32 MRa, mem32
MSUBF32 MRd, MRe, MRf || MMOV32 mem32, MRa
MMPYF32 MRa, MRb, MRc || MSUBF32 MRd, MRe, MRf



Copyright © 2024 Texas Instruments Incorporated

**MSUBF32 MRa, #16FHi, MRb**

***32-Bit Floating-Point Subtraction***

## Operands

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to R1) |
| #16FHi | A 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The lower 16-bits of the mantissa are assumed to be all 0. |
| MRb | CLA floating-point source register (MR0 to R1) |

## Opcode

```
LSW: IIII IIII IIII IIII
MSW: 0111 1000 0000 baaa
```

## Description

Subtract MRb from the floating-point value represented by the immediate operand. Store the result of the addition in MRa.

#16FHi is a 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The lower 16-bits of the mantissa are assumed to be all 0. #16FHi is most useful for representing constants where the lowest 16-bits of the mantissa are 0. Some examples are 2.0 (0x40000000), 4.0 (0x40800000), 0.5 (0x3F000000), and -1.5 (0xBFC00000). The assembler accepts either a hex or float as the immediate value. That is, the value -1.5 can be represented as #-1.5 or #0xBFC0.

```
MRa = #16FHi:0 - MRb;
```

## Flags

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | Yes | Yes |

The MSTF register flags are modified as follows:

- LUF = 1 if MSUBF32 generates an underflow condition.
- LVF = 1 if MSUBF32 generates an overflow condition.

## Pipeline

This is a single-cycle instruction.



Copyright © 2024 Texas Instruments Incorporated

**MSUBF32 MRa, #16FHi, MRb** (continued)

### *32-Bit Floating-Point Subtraction*

**Example**

```
; Y = sqrt(X)
; Ye = Estimate(1/sqrt(X));
; Ye = Ye*(1.5 - Ye*Ye*X*0.5)
; Ye = Ye*(1.5 - Ye*Ye*X*0.5)
; Y = X*Ye
;
_Cla1Task3:
   MMOV32     MR0, @_x           ; MR0 = X
   MEISQRTF32 MR1, MR0           ; MR1 = Ye = Estimate(1/sqrt(X))
   MMOV32     MR1, @_x, EQ       ; if(X == 0.0) Ye = 0.0
   MMPYF32    MR3, MR0, #0.5     ; MR3 = X*0.5
   MMPYF32    MR2, MR1, MR3      ; MR2 = Ye*X*0.5
   MMPYF32    MR2, MR1, MR2      ; MR2 = Ye*Ye*X*0.5
   MSUBF32    MR2, #1.5, MR2     ; MR2 = 1.5 - Ye*Ye*X*0.5
   MMPYF32    MR1, MR1, MR2      ; MR1 = Ye = Ye*(1.5 - Ye*Ye*X*0.5)
   MMPYF32    MR2, MR1, MR3      ; MR2 = Ye*X*0.5
   MMPYF32    MR2, MR1, MR2      ; MR2 = Ye*Ye*X*0.5
   MSUBF32    MR2, #1.5, MR2     ; MR2 = 1.5 - Ye*Ye*X*0.5
   MMPYF32    MR1, MR1, MR2      ; MR1 = Ye = Ye*(1.5 - Ye*Ye*X*0.5)
   MMPYF32    MR0, MR1, MR0      ; MR0 = Y = Ye*X
   MMOV32     @_y, MR0           ; Store Y = sqrt(X)
   MSTOP                         ; end of task
```

**See also**

MSUBF32 MRa, MRb, MRc
MSUBF32 MRd, MRe, MRf || MMOV32 MRa, mem32
MSUBF32 MRd, MRe, MRf || MMOV32 mem32, MRa
MMPYF32 MRa, MRb, MRc || MSUBF32 MRd, MRe, MRf



Copyright © 2024 Texas Instruments Incorporated

**MSUBF32 MRd, MRe, MRf ||MMOV32 MRa, mem32**

***32-Bit Floating-Point Subtraction with Parallel Move***

**Operands**

| | |
|---|---|
| MRd | CLA floating-point destination register (MR0 to MR3) for the MSUBF32 operation<br>MRd cannot be the same register as MRa |
| MRe | CLA floating-point source register (MR0 to MR3) for the MSUBF32 operation |
| MRf | CLA floating-point source register (MR0 to MR3) for the MSUBF32 operation |
| MRa | CLA floating-point destination register (MR0 to MR3) for the MMOV32 operation<br>MRa cannot be the same register as MRd |
| mem32 | 32-bit memory location accessed using one of the available addressing modes. Source for the MMOV32 operation. |

**Opcode**

```
LSW: mmmm mmmm mmmm mmmm
MSW: 0010 ffee ddaa addr
```

**Description**

Subtract the contents of two floating-point registers and move from memory to a floating-point register.

```
MRd = MRe - MRf;
MRa = [mem32];
```

**Restrictions**

The destination register for the MSUBF32 and the MMOV32 must be unique. That is, MRa cannot be the same register as MRd.

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | Yes | Yes |

The MSTF register flags are modified as follows:

- LUF = 1 if MSUBF32 generates an underflow condition.
- LVF = 1 if MSUBF32 generates an overflow condition.

The MMOV32 instruction sets the NF and ZF flags.

**Pipeline**

Both MSUBF32 and MMOV32 complete in a single cycle.

**Example**

```
NF = MRa(31);
ZF = 0;
if(MRa(30:23) == 0) { ZF = 1; NF = 0; }
```

**See also**

MSUBF32 MRa, MRb, MRc
MSUBF32 MRa, #16FHi, MRb
MMPYF32 MRa, MRb, MRc || MSUBF32 MRd, MRe, MRf



Copyright © 2024 Texas Instruments Incorporated

**MSUBF32 MRd, MRe, MRf ||MMOV32 mem32, MRa**

***32-Bit Floating-Point Subtraction with Parallel Move***

## Operands

| | |
|---|---|
| MRd | CLA floating-point destination register (MR0 to MR3) for the MSUBF32 operation |
| MRe | CLA floating-point source register (MR0 to MR3) for the MSUBF32 operation |
| MRf | CLA floating-point source register (MR0 to MR3) for the MSUBF32 operation |
| mem32 | 32-bit destination memory location for the MMOV32 operation |
| MRa | CLA floating-point source register (MR0 to MR3) for the MMOV32 operation |

## Opcode

```
LSW: mmmm mmmm mmmm mmmm
MSW: 0110 ffee ddaa addr
```

## Description

Subtract the contents of two floating-point registers and move from a floating-point register to memory.

```
MRd = MRe - MRf;
[mem32] = MRa;
```

## Flags

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | Yes | Yes |

The MSTF register flags are modified as follows:

- LUF = 1 if MSUBF32 generates an underflow condition.
- LVF = 1 if MSUBF32 generates an overflow condition.

## Pipeline

Both MSUBF32 and MMOV32 complete in a single cycle.

## See also

MSUBF32 MRa, MRb, MRc
MSUBF32 MRa, #16FHi, MRb
MSUBF32 MRd, MRe, MRf || MMOV32 MRa, mem32
MMPYF32 MRa, MRb, MRc || MSUBF32 MRd, MRe, MRf



Copyright © 2024 Texas Instruments Incorporated

**MSWAPF MRa, MRb {, CNDF}**

***Conditional Swap***

**Operands**

| | |
|---|---|
| MRa | CLA floating-point register (MR0 to MR3) |
| MRb | CLA floating-point register (MR0 to MR3) |
| CNDF | Optional condition tested based on the MSTF flags |

**Opcode**

```
LSW: 0000 0000 CNDF bbaa
MSW: 0111 1011 0000 0000
```

**Description**

Conditional swap of MRa and MRb.

```
if (CNDF == true) swap MRa and MRb;
```

CNDF is one of the following conditions:

| Encode[1] | CNDF | Description | MSTF Flags Tested |
|---|---|---|---|
| 0000 | NEQ | Not equal to zero | ZF == 0 |
| 0001 | EQ | Equal to zero | ZF == 1 |
| 0010 | GT | Greater than zero | ZF == 0 AND NF == 0 |
| 0011 | GEQ | Greater than or equal to zero | NF == 0 |
| 0100 | LT | Less than zero | NF == 1 |
| 0101 | LEQ | Less than or equal to zero | ZF == 1 OR NF == 1 |
| 1010 | TF | Test flag set | TF == 1 |
| 1011 | NTF | Test flag not set | TF == 0 |
| 1100 | LU | Latched underflow | LUF == 1 |
| 1101 | LV | Latched overflow | LVF == 1 |
| 1110 | UNC | Unconditional | None |
| 1111 | UNCF[2] | Unconditional with flag modification | None |

(1) Values not shown are reserved.
(2) This is the default operation, if no CNDF field is specified. This condition allows the ZF and NF flags to be modified when a conditional operation is executed. All other conditions do not modify these flags.

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

No flags affected

**Pipeline**

This is a single-cycle instruction.



Copyright © 2024 Texas Instruments Incorporated

**MSWAPF MRa, MRb {, CNDF}** (continued)

### *Conditional Swap*

**Example**

```
; X is an array of 32-bit floating-point values
; and has length elements. Find the maximum value in
; the array and store the value in Result
;
; Note: MCMPF32 and MSWAPF can be replaced by MMAXF32
;
_Cla1Task1:
    MMOVI16    MAR1,#_X         ; Start address
    MUI16TOF32 MR0, @_len       ; Length of the array
    MNOP                        ; delay for MAR1 load
    MNOP                        ; delay for MAR1 load
    MMOV32     MR1, *MAR1[2]++  ; MR1 = X0
LOOP
    MMOV32     MR2, *MAR1[2]++  ; MR2 = next element
    MCMPF32    MR2, MR1         ; Compare MR2 with MR1
    MSWAPF     MR1, MR2, GT     ; MR1 = MAX(MR1, MR2)
    MADDF32    MR0, MR0, #-1.0  ; Decrememt the counter
    MCMPF32    MR0 #0.0         ; Set/clear flags for MBCNDD
    MNOP
    MNOP
    MNOP
    MBCNDD     LOOP, NEQ        ; Branch if not equal to zero
    MMOV32     @_Result, MR1    ; Always executed
    MNOP                        ; Always executed
    MNOP                        ; Always executed
    MSTOP                       ; End of task
```



Copyright © 2024 Texas Instruments Incorporated

## MTESTTF CNDF

### *Test MSTF Register Flag Condition*

**Operands**

| | |
|---|---|
| CNDF | Condition to test based on MSTF flags |

**Opcode**

```
LSW: 0000 0000 0000 cndf
MSW: 0111 1111 0100 0000
```

**Description**

Test the CLA floating-point condition and if true, set the MSTF[TF] flag. If the condition is false, clear the MSTF[TF] flag. This is useful for temporarily storing a condition for later use.

```
if (CNDF == true) TF = 1;
else TF = 0;
```

CNDF is one of the following conditions:

| Encode(1) | CNDF | Description | MSTF Flags Tested |
|---|---|---|---|
| 0000 | NEQ | Not equal to zero | ZF == 0 |
| 0001 | EQ | Equal to zero | ZF == 1 |
| 0010 | GT | Greater than zero | ZF == 0 AND NF == 0 |
| 0011 | GEQ | Greater than or equal to zero | NF == 0 |
| 0100 | LT | Less than zero | NF == 1 |
| 0101 | LEQ | Less than or equal to zero | ZF == 1 OR NF == 1 |
| 1010 | TF | Test flag set | TF == 1 |
| 1011 | NTF | Test flag not set | TF == 0 |
| 1100 | LU | Latched underflow | LUF == 1 |
| 1101 | LV | Latched overflow | LVF == 1 |
| 1110 | UNC | Unconditional | None |
| 1111 | UNCF(2) | Unconditional with flag modification | None |

(1) Values not shown are reserved.
(2) This is the default operation, if no CNDF field is specified. This condition allows the ZF and NF flags to be modified when a conditional operation is executed. All other conditions do not modify these flags.

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | Yes | No | No | No | No |

```
TF = 0;
if (CNDF == true) TF = 1;
```

Note: If (CNDF == UNC or UNCF), the TF flag is set to 1.

**Pipeline**

This is a single-cycle instruction.



Copyright © 2024 Texas Instruments Incorporated

**MTESTTF CNDF** (continued)

***Test MSTF Register Flag Condition***

**Example**

```
; if (State == 0.1)
;    RampState = RampState || RAMPMASK
; else if (State == 0.01)
;    CoastState = CoastState || COASTMASK
; else
;    SteadyState = SteadyState || STEADYMASK
;
_Cla1Task2:
   MMOV32    MR0, @_State
   MCMPF32   MR0, #0.1         ; Affects flags for 1st MBCNDD (A)
   MCMPF32   MR0, #0.01        ; Check used by 2nd MBCNDD (B)
   MTESTTF   EQ                ; Store EQ flag in TF for 2nd MBCNDD (B)
   MNOP
   MBCNDD    _Skip1, NEQ       ; (A) If State != 0.1, go to Skip1
   MMOV32    MR1, @_RampState  ; Always executed
   MMOVXI    MR2, #RAMPMASK    ; Always executed
   MOR32     MR1, MR2          ; Always executed
   MMOV32    @_RampState, MR1  ; Execute if (A) branch not taken
   MSTOP                       ; end of task if (A) branch not taken
_Skip1:
   MMOV32    MR3, @_SteadyState
   MMOVXI    MR2, #STEADYMASK
   MOR32     MR3, MR2
   MBCNDD    _Skip2, NTF       ; (B) if State != .01, go to Skip2
   MMOV32    MR1, @_CoastState ; Always executed
   MMOVXI    MR2, #COASTMASK   ; Always executed
   MOR32     MR1, MR2          ; Always executed
   MMOV32    @_CoastState, MR1 ; Execute if (B) branch not taken
   MSTOP                       ; end of task if (B) branch not taken
_Skip2:
   MMOV32 @_SteadyState, MR3   ; Executed if (B) branch taken
   MSTOP
```



Copyright © 2024 Texas Instruments Incorporated

### MUI16TOF32 MRa, mem16

***Convert Unsigned 16-Bit Integer to 32-Bit Floating-Point Value***

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| mem16 | 16-bit source memory location |

**Opcode**

```
LSW: mmmm mmmm mmmm mmmm
MSW: 0111 0101 01aa addr
```

**Description**

When converting F32 to I16/UI16 data format, the MF32TOI16/UI16 operation truncates to zero while the MF32TOI16R/UI16R operation rounds to the nearest (even) value.

```
MRa = UI16TOF32[mem16];
```

**Flags**

This instruction does not affect any flags:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.

**See also**

MF32TOI16 MRa, MRb
MF32TOI16R MRa, MRb
MF32TOUI16 MRa, MRb
MF32TOUI16R MRa, MRb
MI16TOF32 MRa, MRb
MI16TOF32 MRa, mem16
MUI16TOF32 MRa, MRb



Copyright © 2024 Texas Instruments Incorporated

### MUI16TOF32 MRa, MRb

***Convert Unsigned 16-Bit Integer to 32-Bit Floating-Point Value***

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1110 1110 0000
```

**Description**

Convert an unsigned 16-bit integer to a 32-bit floating-point value. When converting float32 to I16/UI16 data format, the MF32TOI16/UI16 operation truncates to zero while the MF32TOI16R/UI16R operation rounds to the nearest (even) value.

```
MRa = UI16TOF32[MRb];
```

**Flags**

This instruction does not affect any flags:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.

**Example**

```
MMOVXI MR1, #0x800F ; MR1(15:0) = 32783 (0x800F)
MUI16TOF32 MR0, MR1 ; MR0 = UI16TOF32 (MR1(15:0))
                    ; = 32783.0 (0x47000F00)
```

**See also**

MF32TOI16 MRa, MRb
MF32TOI16R MRa, MRb
MF32TOUI16 MRa, MRb
MF32TOUI16R MRa, MRb
MI16TOF32 MRa, MRb
MI16TOF32 MRa, mem16
MUI16TOF32 MRa, mem16



Copyright © 2024 Texas Instruments Incorporated

## MUI32TOF32 MRa, mem32

### *Convert Unsigned 32-Bit Integer to 32-Bit Floating-Point Value*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| mem32 | 32-bit memory location accessed using one of the available addressing modes |

**Opcode**

```
LSW: mmmm mmmm mmmm mmmm
MSW: 0111 0100 10aa addr
```

**Description**

```
MRa = UI32TOF32[mem32];
```

**Flags**

This instruction does not affect any flags:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Given x2, m2, and b2 are Uint32 numbers:
;
; x2 = Uint32(2) = 0x00000002
; m2 = Uint32(1) = 0x00000001
; b2 = Uint32(3) = 0x00000003
;
; Calculate y2 = x2 * m2 + b2
;
_Cla1Task1:
   MUI32TOF32  MR0, @_m2      ; MR0 = 1.0 (0x3F800000)
   MUI32TOF32  MR1, @_x2      ; MR1 = 2.0 (0x40000000)
   MUI32TOF32  MR2, @_b2      ; MR2 = 3.0 (0x40400000)
   MMPYF32     MR3, MR0, MR1  ; M*X
   MADDF32     MR3, MR2, MR3  ; Y=MX+B = 5.0 (0x40A00000)
   MF32TOUI32  MR3, MR3       ; Y = Uint32(5.0) = 0x00000005
   MMOV32      @_y2, MR3      ; store result
   MSTOP                      ; end of task
```

**See also**

MF32TOI32 MRa, MRb
MF32TOUI32 MRa, MRb
MI32TOF32 MRa, mem32
MI32TOF32 MRa, MRb
MUI32TOF32 MRa, MRb



Copyright © 2024 Texas Instruments Incorporated

### MUI32TOF32 MRa, MRb

***Convert Unsigned 32-Bit Integer to 32-Bit Floating-Point Value***

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1101 1100 0000
```

**Description**

```
MRa = UI32TOF32 [MRb];
```

**Flags**

This instruction does not affect any flags:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.

**Example**

```
MMOVIZ     MR3, #0x8000 ; MR3(31:16) = 0x8000
MMOVXI     MR3, #0x1111 ; MR3(15:0) = 0x1111
                        ; MR3 = 2147488017
MUI32TOF32 MR3, MR3     ; MR3 = MUI32TOF32 (MR3) = 2147488017.0 (0x4F000011)
```

**See also**

MF32TOI32 MRa, MRb
MF32TOUI32 MRa, MRb
MI32TOF32 MRa, mem32
MI32TOF32 MRa, MRb
MUI32TOF32 MRa, mem32



Copyright © 2024 Texas Instruments Incorporated

## MXOR32 MRa, MRb, MRc

***Bitwise Exclusive Or***

### Operands

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |
| MRc | CLA floating-point source register (MR0 to MR3) |

### Opcode

```
LSW: 0000 0000 00cc bbaa
MSW: 0111 1100 1010 0000
```

### Description

Bitwise XOR of MRb with MRc.

```
MARa(31:0) = MARb(31:0) XOR MRc(31:0);
```

### Flags

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | No | No |

The MSTF register flags are modified based on the integer results of the operation.

```
NF = MRa(31);
ZF = 0;
if(MRa(31:0) == 0) { ZF = 1; }
```

### Pipeline

This is a single-cycle instruction.

### Example

```
MMOVIZ MR0, #0x5555   ; MR0 = 0x5555AAAA
  MMOVXI MR0, #0xAAAA
  MMOVIZ MR1, #0x5432   ; MR1 = 0x5432FEDC
  MMOVXI MR1, #0xFEDC
  ; 0101 XOR 0101 = 0000 (0)
  ; 0101 XOR 0100 = 0001 (1)
  ; 0101 XOR 0011 = 0110 (6)
  ; 0101 XOR 0010 = 0111 (7)
  ; 1010 XOR 1111 = 0101 (5)
  ; 1010 XOR 1110 = 0100 (4)
  ; 1010 XOR 1101 = 0111 (7)
  ; 1010 XOR 1100 = 0110 (6)
  MXOR32 MR2, MR1, MR0  ; MR3 = 0x01675476
```

### See also

MAND32 MRa, MRb, MRc
MOR32 MRa, MRb, MRc



Copyright © 2024 Texas Instruments Incorporated

## 6.8 CLA Registers

This section describes the Control Law Accelerator registers.

### *6.8.1 CLA Base Addresses*

**Table 6-21. CLA Base Address Table**

| Device Register | Register Name | Start Address | End Address |
|---|---|---|---|
| Cla1Regs | CLA_REGS | 0x0000_1400 | 0x0000_147F |
| Cla1SoftIntRegs(1) | CLA_SOFTINT_REGS | 0x0000_0CE0 | 0x0000_0CFF |

(1) This register is only accessible from the CLA.



Copyright © 2024 Texas Instruments Incorporated

### 6.8.2 CLA_REGS Registers

Table 6-22 lists the memory-mapped registers for the CLA_REGS registers. All register offset addresses not listed in Table 6-22 should be considered as reserved locations and the register contents should not be modified.

**Table 6-22. CLA_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | MVECT1 | Task Interrupt Vector | EALLOW | Go |
| 1h | MVECT2 | Task Interrupt Vector | EALLOW | Go |
| 2h | MVECT3 | Task Interrupt Vector | EALLOW | Go |
| 3h | MVECT4 | Task Interrupt Vector | EALLOW | Go |
| 4h | MVECT5 | Task Interrupt Vector | EALLOW | Go |
| 5h | MVECT6 | Task Interrupt Vector | EALLOW | Go |
| 6h | MVECT7 | Task Interrupt Vector | EALLOW | Go |
| 7h | MVECT8 | Task Interrupt Vector | EALLOW | Go |
| 10h | MCTL | Control Register | EALLOW | Go |
| 20h | MIFR | Interrupt Flag Register | EALLOW | Go |
| 21h | MIOVF | Interrupt Overflow Flag Register | EALLOW | Go |
| 22h | MIFRC | Interrupt Force Register | EALLOW | Go |
| 23h | MICLR | Interrupt Flag Clear Register | EALLOW | Go |
| 24h | MICLROVF | Interrupt Overflow Flag Clear Register | EALLOW | Go |
| 25h | MIER | Interrupt Enable Register | EALLOW | Go |
| 26h | MIRUN | Interrupt Run Status Register | EALLOW | Go |
| 28h | _MPC | CLA Program Counter | | Go |
| 2Ah | _MAR0 | CLA Auxiliary Register 0 | | Go |
| 2Bh | _MAR1 | CLA Auxiliary Register 1 | | Go |
| 2Eh | _MSTF | CLA Floating-Point Status Register | | Go |
| 30h | _MR0 | CLA Floating-Point Result Register 0 | | Go |
| 34h | _MR1 | CLA Floating-Point Result Register 1 | | Go |
| 38h | _MR2 | CLA Floating-Point Result Register 2 | | Go |
| 3Ch | _MR3 | CLA Floating-Point Result Register 3 | | Go |

Complex bit access types are encoded to fit into small table cells. Table 6-23 shows the codes that are used for access types in this section.

**Table 6-23. CLA_REGS Access Type Codes**

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



Copyright © 2024 Texas Instruments Incorporated

**Table 6-23. CLA_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. |



Copyright © 2024 Texas Instruments Incorporated

#### 6.8.2.1 MVECT1 Register (Offset = 0h) [Reset = 0000h]

MVECT1 is shown in Figure 6-2 and described in Table 6-24.

Return to the Summary Table.

Each CLA interrupt has its own interrupt vector (MVECT1 to MVECT8). This interrupt vector points to the first instruction of the associated task. When a task begins, the CLA will start fetching instructions at the location indicated by the appropriate MVECT register .

**Figure 6-2. MVECT1 Register**

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

**Table 6-24. MVECT1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | MVECT | R/W | 0h | MPC Start Address: These bits specify the start address for the given interrupt (task). The address range of the CLA with a 16-bit MVECT is 64Kx16 words or 32K CLA instructions.<br>There is one MVECT register per interrupt (task). Interrupt 1 uses MVECT1, interrupt 2 uses MVECT2 and so forth.<br>Note: While the CLA is running or executing a task, the CPU can change the MVECT values..<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

#### 6.8.2.2 MVECT2 Register (Offset = 1h) [Reset = 0000h]

MVECT2 is shown in Figure 6-3 and described in Table 6-25.

Return to the Summary Table.

Each CLA interrupt has its own interrupt vector (MVECT1 to MVECT8). This interrupt vector points to the first instruction of the associated task. When a task begins, the CLA will start fetching instructions at the location indicated by the appropriate MVECT register .

**Figure 6-3. MVECT2 Register**

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

**Table 6-25. MVECT2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | MVECT | R/W | 0h | MPC Start Address: These bits specify the start address for the given interrupt (task). The address range of the CLA with a 16-bit MVECT is 64Kx16 words or 32K CLA instructions.<br>There is one MVECT register per interrupt (task). Interrupt 1 uses MVECT1, interrupt 2 uses MVECT2 and so forth.<br>Note: While the CLA is running or executing a task, the CPU can change the MVECT values..<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 6.8.2.3 MVECT3 Register (Offset = 2h) [Reset = 0000h]

MVECT3 is shown in Figure 6-4 and described in Table 6-26.

Return to the Summary Table.

Each CLA interrupt has its own interrupt vector (MVECT1 to MVECT8). This interrupt vector points to the first instruction of the associated task. When a task begins, the CLA will start fetching instructions at the location indicated by the appropriate MVECT register .

**Figure 6-4. MVECT3 Register**

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

**Table 6-26. MVECT3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | MVECT | R/W | 0h | MPC Start Address: These bits specify the start address for the given interrupt (task). The address range of the CLA with a 16-bit MVECT is 64Kx16 words or 32K CLA instructions.<br>There is one MVECT register per interrupt (task). Interrupt 1 uses MVECT1, interrupt 2 uses MVECT2 and so forth.<br>Note: While the CLA is running or executing a task, the CPU can change the MVECT values..<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 6.8.2.4 MVECT4 Register (Offset = 3h) [Reset = 0000h]

MVECT4 is shown in Figure 6-5 and described in Table 6-27.

Return to the Summary Table.

Each CLA interrupt has its own interrupt vector (MVECT1 to MVECT8). This interrupt vector points to the first instruction of the associated task. When a task begins, the CLA will start fetching instructions at the location indicated by the appropriate MVECT register .

**Figure 6-5. MVECT4 Register**

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

**Table 6-27. MVECT4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | MVECT | R/W | 0h | MPC Start Address: These bits specify the start address for the given interrupt (task). The address range of the CLA with a 16-bit MVECT is 64Kx16 words or 32K CLA instructions.<br>There is one MVECT register per interrupt (task). Interrupt 1 uses MVECT1, interrupt 2 uses MVECT2 and so forth.<br>Note: While the CLA is running or executing a task, the CPU can change the MVECT values..<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 6.8.2.5 MVECT5 Register (Offset = 4h) [Reset = 0000h]

MVECT5 is shown in Figure 6-6 and described in Table 6-28.

Return to the Summary Table.

Each CLA interrupt has its own interrupt vector (MVECT1 to MVECT8). This interrupt vector points to the first instruction of the associated task. When a task begins, the CLA will start fetching instructions at the location indicated by the appropriate MVECT register .

**Figure 6-6. MVECT5 Register**

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

**Table 6-28. MVECT5 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | MVECT | R/W | 0h | MPC Start Address: These bits specify the start address for the given interrupt (task). The address range of the CLA with a 16-bit MVECT is 64Kx16 words or 32K CLA instructions.<br>There is one MVECT register per interrupt (task). Interrupt 1 uses MVECT1, interrupt 2 uses MVECT2 and so forth.<br>Note: While the CLA is running or executing a task, the CPU can change the MVECT values..<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 6.8.2.6 MVECT6 Register (Offset = 5h) [Reset = 0000h]

MVECT6 is shown in Figure 6-7 and described in Table 6-29.

Return to the Summary Table.

Each CLA interrupt has its own interrupt vector (MVECT1 to MVECT8). This interrupt vector points to the first instruction of the associated task. When a task begins, the CLA will start fetching instructions at the location indicated by the appropriate MVECT register .

**Figure 6-7. MVECT6 Register**

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

**Table 6-29. MVECT6 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | MVECT | R/W | 0h | MPC Start Address: These bits specify the start address for the given interrupt (task). The address range of the CLA with a 16-bit MVECT is 64Kx16 words or 32K CLA instructions.<br>There is one MVECT register per interrupt (task). Interrupt 1 uses MVECT1, interrupt 2 uses MVECT2 and so forth.<br>Note: While the CLA is running or executing a task, the CPU can change the MVECT values..<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 6.8.2.7 MVECT7 Register (Offset = 6h) [Reset = 0000h]

MVECT7 is shown in Figure 6-8 and described in Table 6-30.

Return to the Summary Table.

Each CLA interrupt has its own interrupt vector (MVECT1 to MVECT8). This interrupt vector points to the first instruction of the associated task. When a task begins, the CLA will start fetching instructions at the location indicated by the appropriate MVECT register .

**Figure 6-8. MVECT7 Register**

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

**Table 6-30. MVECT7 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | MVECT | R/W | 0h | MPC Start Address: These bits specify the start address for the given interrupt (task). The address range of the CLA with a 16-bit MVECT is 64Kx16 words or 32K CLA instructions.<br>There is one MVECT register per interrupt (task). Interrupt 1 uses MVECT1, interrupt 2 uses MVECT2 and so forth.<br>Note: While the CLA is running or executing a task, the CPU can change the MVECT values..<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 6.8.2.8 MVECT8 Register (Offset = 7h) [Reset = 0000h]

MVECT8 is shown in Figure 6-9 and described in Table 6-31.

Return to the Summary Table.

Each CLA interrupt has its own interrupt vector (MVECT1 to MVECT8). This interrupt vector points to the first instruction of the associated task. When a task begins, the CLA will start fetching instructions at the location indicated by the appropriate MVECT register .

**Figure 6-9. MVECT8 Register**

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

**Table 6-31. MVECT8 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | MVECT | R/W | 0h | MPC Start Address: These bits specify the start address for the given interrupt (task). The address range of the CLA with a 16-bit MVECT is 64Kx16 words or 32K CLA instructions.<br>There is one MVECT register per interrupt (task). Interrupt 1 uses MVECT1, interrupt 2 uses MVECT2 and so forth.<br>Note: While the CLA is running or executing a task, the CPU can change the MVECT values..<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 6.8.2.9 MCTL Register (Offset = 10h) [Reset = 0000h]

MCTL is shown in Figure 6-10 and described in Table 6-32.

Return to the Summary Table.

Control Register

**Figure 6-10. MCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | IACKE | SOFTRESET | HARDRESET |
| R-0h | | | | | R/W-0h | R-0/W1S-0h | R-0/W1S-0h |

**Table 6-32. MCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-3 | RESERVED | R | 0h | Reserved |
| 2 | IACKE | R/W | 0h | IACK Operation Enable Bit: Writing a '1' to this bit will enable the IACK operation for setting the MIFR bits in the same manner as the MIFRC register (write of '1' will set respective MIFR bit). At reset, this feature is disabled.<br>This feature enables the C28 CPU to efficiently trigger a task.<br>Note: IACK operation should ignore EALLOW status of C28 core when accessing the MIFRC register.<br>Reset type: SYSRSn<br>0h (R/W) = The CLA ignores the IACK instruction. (default)<br>1h (R/W) = Enable the main CPU to use the IACK #16bit instruction to set MIFR bits in the same manner as writing to the MIFRC register. Each bit in the operand, #16bit, corresponds to a bit in the MIFRC register. Using IACK has the advantage of not having to first set the EALLOW bit. This allows the main CPU to efficiently trigger a CLA task through software.<br>Examples IACK #0x0001 Write a 1 to MIFRC bit 0 to force task 1<br>IACK #0x0003 Write a 1 to MIFRC bit 0 and 1 to force task 1 and task 2 |
| 1 | SOFTRESET | R-0/W1S | 0h | Soft Reset Bit: Writing a '1' to this bit will stop a current task, clear the RUN flag and also clear all bits in the MIER register. Writes of '0' are ignored and reads always return a '0'.<br>Note: After issuing SOFTRESET command, user should wait at least 1 clock cycle before attempting to write to MIER register.<br>Reset type: SYSRSn<br>0h (R/W) = This bit always reads back 0 and writes of 0 are ignored.<br>1h (R/W) = Writing a 1 will cause a soft reset of the CLA. This will stop the current task, clear the MIRUN flag and clear all bits in the MIER register. After a soft reset you must wait at least 1 SYSCLKOUT cycle before reconfiguring the MIER bits. If these two operations are done back-to-back then the MIER bits will not get set. |
| 0 | HARDRESET | R-0/W1S | 0h | Hard Reset Bit: Writing a '1' to this bit will cause a HARD reset on the CLA. The behavior of a HARD reset is the same as a system reset SYSRSn on the CLA. Writes of '0' are ignored and reads always return a '0'.<br>Reset type: SYSRSn<br>0h (R/W) = This bit always reads back 0 and writes of 0 are ignored.<br>1h (R/W) = Writing a 1 will cause a hard reset of the CLA. This will set all CLA registers to their default state. |



Copyright © 2024 Texas Instruments Incorporated

### 6.8.2.10 MIFR Register (Offset = 20h) [Reset = 0000h]

MIFR is shown in Figure 6-11 and described in Table 6-33.

Return to the Summary Table.

Each bit in the interrupt flag register corresponds to a CLA task. The corresponding bit is automatically set when the task request is received from the peripheral interrupt. The bit can also be set by the main CPU writing to the MIFRC register or using the IACK instruction to start the task. To use the IACK instruction to begin a task first enable this feature in the MCTL register. If the bit is already set when a new peripheral interrupt is received, then the corresponding overflow bit will be set in the MIOVF register.
The corresponding MIFR bit is automatically cleared when the task begins execution. This will occur if the interrupt is enabled in the MIER register and no other higher priority task is pending. The bits can also be cleared manually by writing to the MICLR register. Writes to the MIFR register are ignored.

**Figure 6-11. MIFR Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INT8 | INT7 | INT6 | INT5 | INT4 | INT3 | INT2 | INT1 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 6-33. MIFR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R | 0h | Reserved |
| 7 | INT8 | R | 0h | These bits, when set to '1', indicate a valid peripheral interrupt has been latched by the CLA. Writes to this register are ignored.<br>The IFR flag bit is automatically cleared if the respective interrupt is enabled in the MIER register and the respective task starts running. If a new peripheral interrupt attempts to set the bit to '1' while on the same cycle the task tries to clear it, then the peripheral interrupt will have priority.<br>The IFR flag bits can also be set and cleared by the MIFRC and MICLR registers.<br>If the MIFRC register is trying to set the respective bit while a new task tries to clear it, then the MIFRC event has priority.<br>If the MICLR register is trying to clear the respective bit and a peripheral interrupt occurs on the same cycle, then the peripheral interrupt has priority. The respective overflow flag in the MIOVF register will not be set under this condition.<br>Reset type: SYSRSn<br>0h (R/W) = TASK_FLAG_DISABLE<br>Task 8 interrupt is currently not flagged (default)<br>1h (R/W) = TASK_FLAG_ENABLE<br>Task 8 interrupt has been received and is pending execution |



Copyright © 2024 Texas Instruments Incorporated

**Table 6-33. MIFR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 6 | INT7 | R | 0h | These bits, when set to '1', indicate a valid peripheral interrupt has been latched by the CLA. Writes to this register are ignored.<br>The IFR flag bit is automatically cleared if the respective interrupt is enabled in the MIER register and the respective task starts running. If a new peripheral interrupt attempts to set the bit to '1' while on the same cycle the task tries to clear it, then the peripheral interrupt will have priority.<br>The IFR flag bits can also be set and cleared by the MIFRC and MICLR registers.<br>If the MIFRC register is trying to set the respective bit while a new task tries to clear it, then the MIFRC event has priority.<br>If the MICLR register is trying to clear the respective bit and a peripheral interrupt occurs on the same cycle, then the peripheral interrupt has priority. The respective overflow flag in the MIOVF register will not be set under this condition.<br>Reset type: SYSRSn<br>0h (R/W) = TASK_FLAG_DISABLE<br>Task 7 interrupt is currently not flagged (default)<br>1h (R/W) = TASK_FLAG_ENABLE<br>Task 7 interrupt has been received and is pending execution |
| 5 | INT6 | R | 0h | These bits, when set to '1', indicate a valid peripheral interrupt has been latched by the CLA. Writes to this register are ignored.<br>The IFR flag bit is automatically cleared if the respective interrupt is enabled in the MIER register and the respective task starts running. If a new peripheral interrupt attempts to set the bit to '1' while on the same cycle the task tries to clear it, then the peripheral interrupt will have priority.<br>The IFR flag bits can also be set and cleared by the MIFRC and MICLR registers.<br>If the MIFRC register is trying to set the respective bit while a new task tries to clear it, then the MIFRC event has priority.<br>If the MICLR register is trying to clear the respective bit and a peripheral interrupt occurs on the same cycle, then the peripheral interrupt has priority. The respective overflow flag in the MIOVF register will not be set under this condition.<br>Reset type: SYSRSn<br>0h (R/W) = TASK_FLAG_DISABLE<br>Task 6 interrupt is currently not flagged (default)<br>1h (R/W) = TASK_FLAG_ENABLE<br>Task 6 interrupt has been received and is pending execution |
| 4 | INT5 | R | 0h | These bits, when set to '1', indicate a valid peripheral interrupt has been latched by the CLA. Writes to this register are ignored.<br>The IFR flag bit is automatically cleared if the respective interrupt is enabled in the MIER register and the respective task starts running. If a new peripheral interrupt attempts to set the bit to '1' while on the same cycle the task tries to clear it, then the peripheral interrupt will have priority.<br>The IFR flag bits can also be set and cleared by the MIFRC and MICLR registers.<br>If the MIFRC register is trying to set the respective bit while a new task tries to clear it, then the MIFRC event has priority.<br>If the MICLR register is trying to clear the respective bit and a peripheral interrupt occurs on the same cycle, then the peripheral interrupt has priority. The respective overflow flag in the MIOVF register will not be set under this condition.<br>Reset type: SYSRSn<br>0h (R/W) = TASK_FLAG_DISABLE<br>Task 5 interrupt is currently not flagged (default)<br>1h (R/W) = TASK_FLAG_ENABLE<br>Task 5 interrupt has been received and is pending execution |



Copyright © 2024 Texas Instruments Incorporated

**Table 6-33. MIFR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3 | INT4 | R | 0h | These bits, when set to '1', indicate a valid peripheral interrupt has been latched by the CLA. Writes to this register are ignored.<br>The IFR flag bit is automatically cleared if the respective interrupt is enabled in the MIER register and the respective task starts running.<br>If a new peripheral interrupt attempts to set the bit to '1' while on the same cycle the task tries to clear it, then the peripheral interrupt will have priority.<br>The IFR flag bits can also be set and cleared by the MIFRC and MICLR registers.<br>If the MIFRC register is trying to set the respective bit while a new task tries to clear it, then the MIFRC event has priority.<br>If the MICLR register is trying to clear the respective bit and a peripheral interrupt occurs on the same cycle, then the peripheral interrupt has priority. The respective overflow flag in the MIOVF register will not be set under this condition.<br>Reset type: SYSRSn<br>0h (R/W) = TASK_FLAG_DISABLE<br>Task 4 interrupt is currently not flagged (default)<br>1h (R/W) = TASK_FLAG_ENABLE<br>Task 4 interrupt has been received and is pending execution |
| 2 | INT3 | R | 0h | These bits, when set to '1', indicate a valid peripheral interrupt has been latched by the CLA. Writes to this register are ignored.<br>The IFR flag bit is automatically cleared if the respective interrupt is enabled in the MIER register and the respective task starts running.<br>If a new peripheral interrupt attempts to set the bit to '1' while on the same cycle the task tries to clear it, then the peripheral interrupt will have priority.<br>The IFR flag bits can also be set and cleared by the MIFRC and MICLR registers.<br>If the MIFRC register is trying to set the respective bit while a new task tries to clear it, then the MIFRC event has priority.<br>If the MICLR register is trying to clear the respective bit and a peripheral interrupt occurs on the same cycle, then the peripheral interrupt has priority. The respective overflow flag in the MIOVF register will not be set under this condition.<br>Reset type: SYSRSn<br>0h (R/W) = TASK_FLAG_DISABLE<br>Task 3 interrupt is currently not flagged (default)<br>1h (R/W) = TASK_FLAG_ENABLE<br>Task 3 interrupt has been received and is pending execution |
| 1 | INT2 | R | 0h | These bits, when set to '1', indicate a valid peripheral interrupt has been latched by the CLA. Writes to this register are ignored.<br>The IFR flag bit is automatically cleared if the respective interrupt is enabled in the MIER register and the respective task starts running.<br>If a new peripheral interrupt attempts to set the bit to '1' while on the same cycle the task tries to clear it, then the peripheral interrupt will have priority.<br>The IFR flag bits can also be set and cleared by the MIFRC and MICLR registers.<br>If the MIFRC register is trying to set the respective bit while a new task tries to clear it, then the MIFRC event has priority.<br>If the MICLR register is trying to clear the respective bit and a peripheral interrupt occurs on the same cycle, then the peripheral interrupt has priority. The respective overflow flag in the MIOVF register will not be set under this condition.<br>Reset type: SYSRSn<br>0h (R/W) = TASK_FLAG_DISABLE<br>Task 2 interrupt is currently not flagged (default)<br>1h (R/W) = TASK_FLAG_ENABLE<br>Task 2 interrupt has been received and is pending execution |



Copyright © 2024 Texas Instruments Incorporated

**Table 6-33. MIFR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | INT1 | R | 0h | These bits, when set to '1', indicate a valid peripheral interrupt has been latched by the CLA. Writes to this register are ignored.<br>The IFR flag bit is automatically cleared if the respective interrupt is enabled in the MIER register and the respective task starts running.<br>If a new peripheral interrupt attempts to set the bit to '1' while on the same cycle the task tries to clear it, then the peripheral interrupt will have priority.<br>The IFR flag bits can also be set and cleared by the MIFRC and MICLR registers.<br>If the MIFRC register is trying to set the respective bit while a new task tries to clear it, then the MIFRC event has priority.<br>If the MICLR register is trying to clear the respective bit and a peripheral interrupt occurs on the same cycle, then the peripheral interrupt has priority. The respective overflow flag in the MIOVF register will not be set under this condition.<br>Reset type: SYSRSn<br>0h (R/W) = TASK_FLAG_DISABLE<br>Task 1 interrupt is currently not flagged (default)<br>1h (R/W) = TASK_FLAG_ENABLE<br>Task 1 interrupt has been received and is pending execution |



Copyright © 2024 Texas Instruments Incorporated

### 6.8.2.11 MIOVF Register (Offset = 21h) [Reset = 0000h]

MIOVF is shown in Figure 6-12 and described in Table 6-34.

Return to the Summary Table.

Each bit in the overflow flag register corresponds to a CLA task. The bit is set when an interrupt overflow event has occurred for the specific task. An overflow event occurs when the MIFR register bit is already set when a new interrupt is received from a peripheral source. The MIOVF bits are only affected by peripheral interrupt events. They do not respond to a task request by the main CPU IACK instruction or by directly setting MIFR bits. The overflow flag will remain latched and can only be cleared by writing to the overflow flag clear (MICLROVF) register. Writes to the MIOVF register are ignored.

**Figure 6-12. MIOVF Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INT8 | INT7 | INT6 | INT5 | INT4 | INT3 | INT2 | INT1 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 6-34. MIOVF Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R | 0h | Reserved |
| 7 | INT8 | R | 0h | These bits, when set to '1', indicate an interrupt overflow event occurred. Such an event occurs when the IFR bit is already set.<br>An overflow event remains latched and respective bits can only be cleared by writing to the MICLROVF register.<br>If the MIFR bit is being cleared by a new task on the same cycle as a new peripheral interrupt occurs, the overflow flag will not be affected and the respective MIFR bit will be set.<br>If the MIOVF bit is being cleared by the MICLROVF register on the same cycle as the overflow bit is being set by hardware, then the hardware will have priority.<br>Notes: [1] The MIOVF bits are only affected by peripheral interrupt events. Forcing an interrupt using the MIFRC or IACK operation will not set the overflow flag even if the MIFR bit is set.<br>Reset type: SYSRSn<br>0h (R/W) = A task 8 interrupt overflow has not occurred (default)<br>1h (R/W) = A task 8 interrupt overflow has occurred |
| 6 | INT7 | R | 0h | These bits, when set to '1', indicate an interrupt overflow event occurred. Such an event occurs when the IFR bit is already set.<br>An overflow event remains latched and respective bits can only be cleared by writing to the MICLROVF register.<br>If the MIFR bit is being cleared by a new task on the same cycle as a new peripheral interrupt occurs, the overflow flag will not be affected and the respective MIFR bit will be set.<br>If the MIOVF bit is being cleared by the MICLROVF register on the same cycle as the overflow bit is being set by hardware, then the hardware will have priority.<br>Notes: [1] The MIOVF bits are only affected by peripheral interrupt events. Forcing an interrupt using the MIFRC or IACK operation will not set the overflow flag even if the MIFR bit is set.<br>Reset type: SYSRSn<br>0h (R/W) = A task 7 interrupt overflow has not occurred (default)<br>1h (R/W) = A task 7 interrupt overflow has occurred |



Copyright © 2024 Texas Instruments Incorporated

**Table 6-34. MIOVF Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 5 | INT6 | R | 0h | These bits, when set to '1', indicate an interrupt overflow event occurred. Such an event occurs when the IFR bit is already set.<br>An overflow event remains latched and respective bits can only be cleared by writing to the MICLROVF register.<br>If the MIFR bit is being cleared by a new task on the same cycle as a new peripheral interrupt occurs, the overflow flag will not be affected and the respective MIFR bit will be set.<br>If the MIOVF bit is being cleared by the MICLROVF register on the same cycle as the overflow bit is being set by hardware, then the hardware will have priority.<br>Notes: [1] The MIOVF bits are only affected by peripheral interrupt events. Forcing an interrupt using the MIFRC or IACK operation will not set the overflow flag even if the MIFR bit is set.<br>Reset type: SYSRSn<br>0h (R/W) = A task 6 interrupt overflow has not occurred (default)<br>1h (R/W) = A task 6 interrupt overflow has occurred |
| 4 | INT5 | R | 0h | These bits, when set to '1', indicate an interrupt overflow event occurred. Such an event occurs when the IFR bit is already set.<br>An overflow event remains latched and respective bits can only be cleared by writing to the MICLROVF register.<br>If the MIFR bit is being cleared by a new task on the same cycle as a new peripheral interrupt occurs, the overflow flag will not be affected and the respective MIFR bit will be set.<br>If the MIOVF bit is being cleared by the MICLROVF register on the same cycle as the overflow bit is being set by hardware, then the hardware will have priority.<br>Notes: [1] The MIOVF bits are only affected by peripheral interrupt events. Forcing an interrupt using the MIFRC or IACK operation will not set the overflow flag even if the MIFR bit is set.<br>Reset type: SYSRSn<br>0h (R/W) = A task 5 interrupt overflow has not occurred (default)<br>1h (R/W) = A task 5 interrupt overflow has occurred |
| 3 | INT4 | R | 0h | These bits, when set to '1', indicate an interrupt overflow event occurred. Such an event occurs when the IFR bit is already set.<br>An overflow event remains latched and respective bits can only be cleared by writing to the MICLROVF register.<br>If the MIFR bit is being cleared by a new task on the same cycle as a new peripheral interrupt occurs, the overflow flag will not be affected and the respective MIFR bit will be set.<br>If the MIOVF bit is being cleared by the MICLROVF register on the same cycle as the overflow bit is being set by hardware, then the hardware will have priority.<br>Notes: [1] The MIOVF bits are only affected by peripheral interrupt events. Forcing an interrupt using the MIFRC or IACK operation will not set the overflow flag even if the MIFR bit is set.<br>Reset type: SYSRSn<br>0h (R/W) = A task 4 interrupt overflow has not occurred (default)<br>1h (R/W) = A task 4 interrupt overflow has occurred |



Copyright © 2024 Texas Instruments Incorporated

**Table 6-34. MIOVF Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 2 | INT3 | R | 0h | These bits, when set to '1', indicate an interrupt overflow event occurred. Such an event occurs when the IFR bit is already set.<br>An overflow event remains latched and respective bits can only be cleared by writing to the MICLROVF register.<br>If the MIFR bit is being cleared by a new task on the same cycle as a new peripheral interrupt occurs, the overflow flag will not be affected and the respective MIFR bit will be set.<br>If the MIOVF bit is being cleared by the MICLROVF register on the same cycle as the overflow bit is being set by hardware, then the hardware will have priority.<br>Notes: [1] The MIOVF bits are only affected by peripheral interrupt events. Forcing an interrupt using the MIFRC or IACK operation will not set the overflow flag even if the MIFR bit is set.<br>Reset type: SYSRSn<br>0h (R/W) = A task 3 interrupt overflow has not occurred (default)<br>1h (R/W) = A task 3 interrupt overflow has occurred |
| 1 | INT2 | R | 0h | These bits, when set to '1', indicate an interrupt overflow event occurred. Such an event occurs when the IFR bit is already set.<br>An overflow event remains latched and respective bits can only be cleared by writing to the MICLROVF register.<br>If the MIFR bit is being cleared by a new task on the same cycle as a new peripheral interrupt occurs, the overflow flag will not be affected and the respective MIFR bit will be set.<br>If the MIOVF bit is being cleared by the MICLROVF register on the same cycle as the overflow bit is being set by hardware, then the hardware will have priority.<br>Notes: [1] The MIOVF bits are only affected by peripheral interrupt events. Forcing an interrupt using the MIFRC or IACK operation will not set the overflow flag even if the MIFR bit is set.<br>Reset type: SYSRSn<br>0h (R/W) = A task 2 interrupt overflow has not occurred (default)<br>1h (R/W) = A task 2 interrupt overflow has occurred |
| 0 | INT1 | R | 0h | These bits, when set to '1', indicate an interrupt overflow event occurred. Such an event occurs when the IFR bit is already set.<br>An overflow event remains latched and respective bits can only be cleared by writing to the MICLROVF register.<br>If the MIFR bit is being cleared by a new task on the same cycle as a new peripheral interrupt occurs, the overflow flag will not be affected and the respective MIFR bit will be set.<br>If the MIOVF bit is being cleared by the MICLROVF register on the same cycle as the overflow bit is being set by hardware, then the hardware will have priority.<br>Notes: [1] The MIOVF bits are only affected by peripheral interrupt events. Forcing an interrupt using the MIFRC or IACK operation will not set the overflow flag even if the MIFR bit is set.<br>Reset type: SYSRSn<br>0h (R/W) = A task 1 interrupt overflow has not occurred (default)<br>1h (R/W) = A task 1 interrupt overflow has occurred |



Copyright © 2024 Texas Instruments Incorporated

### 6.8.2.12 MIFRC Register (Offset = 22h) [Reset = 0000h]

MIFRC is shown in Figure 6-13 and described in Table 6-35.

Return to the Summary Table.

The interrupt force register can be used by the main CPU to start tasks through software. Writing a 1 to a MIFRC bit will set the corresponding bit in the MIFR register. Writes of 0 are ignored and reads always return 0. The IACK #16bit operation can also be used to start tasks and has the same effect as the MIFRC register. To enable IACK to set MIFR bits you must first set the MCTL[IACKE] bit. Using IACK has the advantage of not having to first set the EALLOW bit. This allows the main CPU to efficiently trigger CLA tasks through software.

**Figure 6-13. MIFRC Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INT8 | INT7 | INT6 | INT5 | INT4 | INT3 | INT2 | INT1 |
| R-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 6-35. MIFRC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R | 0h | Reserved |
| 7 | INT8 | R-0/W1S | 0h | Writing a '1' to any of the bits will set the corresponding MIFR bit. Writes of '0' are ignored. Reads always return 0.<br>Notes: [1] Refer to MIFR register description for handling of boundary conditions.<br>Reset type: SYSRSn<br>0h (R/W) = This bit always reads back 0 and writes of 0 have no effect<br>1h (R/W) = Write a 1 to force the task 8 interrupt |
| 6 | INT7 | R-0/W1S | 0h | Writing a '1' to any of the bits will set the corresponding MIFR bit. Writes of '0' are ignored. Reads always return 0.<br>Notes: [1] Refer to MIFR register description for handling of boundary conditions.<br>Reset type: SYSRSn<br>0h (R/W) = This bit always reads back 0 and writes of 0 have no effect<br>1h (R/W) = Write a 1 to force the task 7 interrupt |
| 5 | INT6 | R-0/W1S | 0h | Writing a '1' to any of the bits will set the corresponding MIFR bit. Writes of '0' are ignored. Reads always return 0.<br>Notes: [1] Refer to MIFR register description for handling of boundary conditions.<br>Reset type: SYSRSn<br>0h (R/W) = This bit always reads back 0 and writes of 0 have no effect<br>1h (R/W) = Write a 1 to force the task 6 interrupt |
| 4 | INT5 | R-0/W1S | 0h | Writing a '1' to any of the bits will set the corresponding MIFR bit. Writes of '0' are ignored. Reads always return 0.<br>Notes: [1] Refer to MIFR register description for handling of boundary conditions.<br>Reset type: SYSRSn<br>0h (R/W) = This bit always reads back 0 and writes of 0 have no effect<br>1h (R/W) = Write a 1 to force the task 5 interrupt |



Copyright © 2024 Texas Instruments Incorporated

**Table 6-35. MIFRC Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3 | INT4 | R-0/W1S | 0h | Writing a '1' to any of the bits will set the corresponding MIFR bit. Writes of '0' are ignored. Reads always return 0.<br>Notes: [1] Refer to MIFR register description for handling of boundary conditions.<br>Reset type: SYSRSn<br>0h (R/W) = This bit always reads back 0 and writes of 0 have no effect<br>1h (R/W) = Write a 1 to force the task 4 interrupt |
| 2 | INT3 | R-0/W1S | 0h | Writing a '1' to any of the bits will set the corresponding MIFR bit. Writes of '0' are ignored. Reads always return 0.<br>Notes: [1] Refer to MIFR register description for handling of boundary conditions.<br>Reset type: SYSRSn<br>0h (R/W) = This bit always reads back 0 and writes of 0 have no effect<br>1h (R/W) = Write a 1 to force the task 3 interrupt |
| 1 | INT2 | R-0/W1S | 0h | Writing a '1' to any of the bits will set the corresponding MIFR bit. Writes of '0' are ignored. Reads always return 0.<br>Notes: [1] Refer to MIFR register description for handling of boundary conditions.<br>Reset type: SYSRSn<br>0h (R/W) = This bit always reads back 0 and writes of 0 have no effect<br>1h (R/W) = Write a 1 to force the task 2 interrupt |
| 0 | INT1 | R-0/W1S | 0h | Writing a '1' to any of the bits will set the corresponding MIFR bit. Writes of '0' are ignored. Reads always return 0.<br>Notes: [1] Refer to MIFR register description for handling of boundary conditions.<br>Reset type: SYSRSn<br>0h (R/W) = This bit always reads back 0 and writes of 0 have no effect<br>1h (R/W) = Write a 1 to force the task 1 interrupt |



Copyright © 2024 Texas Instruments Incorporated

### 6.8.2.13 MICLR Register (Offset = 23h) [Reset = 0000h]

MICLR is shown in Figure 6-14 and described in Table 6-36.

Return to the Summary Table.

Normally bits in the MIFR register are automatically cleared when a task begins. The interrupt flag clear register can be used to instead manually clear bits in the interrupt flag (MIFR) register. Writing a 1 to a MICLR bit will clear the corresponding bit in the MIFR register. Writes of 0 are ignored and reads always return 0.

**Figure 6-14. MICLR Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INT8 | INT7 | INT6 | INT5 | INT4 | INT3 | INT2 | INT1 |
| R-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 6-36. MICLR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R | 0h | Reserved |
| 7 | INT8 | R-0/W1S | 0h | Writing a '1' to any of the bits will clear the corresponding MIFR bit. Writes of '0' are ignored. Reads always return 0.<br>Notes: [1] Refer to MIFR register description for handling of boundary conditions.<br>Reset type: SYSRSn<br>0h (R/W) = This bit always reads back 0 and writes of 0 have no effect<br>1h (R/W) = Write a 1 to clear the task 8 interrupt flag |
| 6 | INT7 | R-0/W1S | 0h | Writing a '1' to any of the bits will clear the corresponding MIFR bit. Writes of '0' are ignored. Reads always return 0.<br>Notes: [1] Refer to MIFR register description for handling of boundary conditions.<br>Reset type: SYSRSn<br>0h (R/W) = This bit always reads back 0 and writes of 0 have no effect<br>1h (R/W) = Write a 1 to clear the task 7 interrupt flag |
| 5 | INT6 | R-0/W1S | 0h | Writing a '1' to any of the bits will clear the corresponding MIFR bit. Writes of '0' are ignored. Reads always return 0.<br>Notes: [1] Refer to MIFR register description for handling of boundary conditions.<br>Reset type: SYSRSn<br>0h (R/W) = This bit always reads back 0 and writes of 0 have no effect<br>1h (R/W) = Write a 1 to clear the task 6 interrupt flag |
| 4 | INT5 | R-0/W1S | 0h | Writing a '1' to any of the bits will clear the corresponding MIFR bit. Writes of '0' are ignored. Reads always return 0.<br>Notes: [1] Refer to MIFR register description for handling of boundary conditions.<br>Reset type: SYSRSn<br>0h (R/W) = This bit always reads back 0 and writes of 0 have no effect<br>1h (R/W) = Write a 1 to clear the task 5 interrupt flag |



Copyright © 2024 Texas Instruments Incorporated

**Table 6-36. MICLR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3 | INT4 | R-0/W1S | 0h | Writing a '1' to any of the bits will clear the corresponding MIFR bit. Writes of '0' are ignored. Reads always return 0.<br>Notes: [1] Refer to MIFR register description for handling of boundary conditions.<br>Reset type: SYSRSn<br>0h (R/W) = This bit always reads back 0 and writes of 0 have no effect<br>1h (R/W) = Write a 1 to clear the task 4 interrupt flag |
| 2 | INT3 | R-0/W1S | 0h | Writing a '1' to any of the bits will clear the corresponding MIFR bit. Writes of '0' are ignored. Reads always return 0.<br>Notes: [1] Refer to MIFR register description for handling of boundary conditions.<br>Reset type: SYSRSn<br>0h (R/W) = This bit always reads back 0 and writes of 0 have no effect<br>1h (R/W) = Write a 1 to clear the task 3 interrupt flag |
| 1 | INT2 | R-0/W1S | 0h | Writing a '1' to any of the bits will clear the corresponding MIFR bit. Writes of '0' are ignored. Reads always return 0.<br>Notes: [1] Refer to MIFR register description for handling of boundary conditions.<br>Reset type: SYSRSn<br>0h (R/W) = This bit always reads back 0 and writes of 0 have no effect<br>1h (R/W) = Write a 1 to clear the task 2 interrupt flag |
| 0 | INT1 | R-0/W1S | 0h | Writing a '1' to any of the bits will clear the corresponding MIFR bit. Writes of '0' are ignored. Reads always return 0.<br>Notes: [1] Refer to MIFR register description for handling of boundary conditions.<br>Reset type: SYSRSn<br>0h (R/W) = This bit always reads back 0 and writes of 0 have no effect<br>1h (R/W) = Write a 1 to clear the task 1 interrupt flag |



Copyright © 2024 Texas Instruments Incorporated

### 6.8.2.14 MICLROVF Register (Offset = 24h) [Reset = 0000h]

MICLROVF is shown in Figure 6-15 and described in Table 6-37.

Return to the Summary Table.

Overflow flag bits in the MIOVF register are latched until manually cleared using the MICLROVF register. Writing a 1 to a MICLROVF bit will clear the corresponding bit in the MIOVF register. Writes of 0 are ignored and reads always return 0.

**Figure 6-15. MICLROVF Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INT8 | INT7 | INT6 | INT5 | INT4 | INT3 | INT2 | INT1 |
| R-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 6-37. MICLROVF Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R | 0h | Reserved |
| 7 | INT8 | R-0/W1S | 0h | Writing a '1' to any of the bits will clear the corresponding MIOVF bit. Writes of '0' are ignored. Reads always return 0.<br>Notes: [1] Refer to MIOVF register description for handling of boundary conditions.<br>Reset type: SYSRSn<br>0h (R/W) = This bit always reads back 0 and writes of 0 have no effect<br>1h (R/W) = Write a 1 to clear the task 8 interrupt overflow flag |
| 6 | INT7 | R-0/W1S | 0h | Writing a '1' to any of the bits will clear the corresponding MIOVF bit. Writes of '0' are ignored. Reads always return 0.<br>Notes: [1] Refer to MIOVF register description for handling of boundary conditions.<br>Reset type: SYSRSn<br>0h (R/W) = This bit always reads back 0 and writes of 0 have no effect<br>1h (R/W) = Write a 1 to clear the task 7 interrupt overflow flag |
| 5 | INT6 | R-0/W1S | 0h | Writing a '1' to any of the bits will clear the corresponding MIOVF bit. Writes of '0' are ignored. Reads always return 0.<br>Notes: [1] Refer to MIOVF register description for handling of boundary conditions.<br>Reset type: SYSRSn<br>0h (R/W) = This bit always reads back 0 and writes of 0 have no effect<br>1h (R/W) = Write a 1 to clear the task 6 interrupt overflow flag |
| 4 | INT5 | R-0/W1S | 0h | Writing a '1' to any of the bits will clear the corresponding MIOVF bit. Writes of '0' are ignored. Reads always return 0.<br>Notes: [1] Refer to MIOVF register description for handling of boundary conditions.<br>Reset type: SYSRSn<br>0h (R/W) = This bit always reads back 0 and writes of 0 have no effect<br>1h (R/W) = Write a 1 to clear the task 5 interrupt overflow flag |



Copyright © 2024 Texas Instruments Incorporated

**Table 6-37. MICLROVF Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3 | INT4 | R-0/W1S | 0h | Writing a '1' to any of the bits will clear the corresponding MIOVF bit. Writes of '0' are ignored. Reads always return 0.<br>Notes: [1] Refer to MIOVF register description for handling of boundary conditions.<br>Reset type: SYSRSn<br>0h (R/W) = This bit always reads back 0 and writes of 0 have no effect<br>1h (R/W) = Write a 1 to clear the task 4 interrupt overflow flag |
| 2 | INT3 | R-0/W1S | 0h | Writing a '1' to any of the bits will clear the corresponding MIOVF bit. Writes of '0' are ignored. Reads always return 0.<br>Notes: [1] Refer to MIOVF register description for handling of boundary conditions.<br>Reset type: SYSRSn<br>0h (R/W) = This bit always reads back 0 and writes of 0 have no effect<br>1h (R/W) = Write a 1 to clear the task 3 interrupt overflow flag |
| 1 | INT2 | R-0/W1S | 0h | Writing a '1' to any of the bits will clear the corresponding MIOVF bit. Writes of '0' are ignored. Reads always return 0.<br>Notes: [1] Refer to MIOVF register description for handling of boundary conditions.<br>Reset type: SYSRSn<br>0h (R/W) = This bit always reads back 0 and writes of 0 have no effect<br>1h (R/W) = Write a 1 to clear the task 2 interrupt overflow flag |
| 0 | INT1 | R-0/W1S | 0h | Writing a '1' to any of the bits will clear the corresponding MIOVF bit. Writes of '0' are ignored. Reads always return 0.<br>Notes: [1] Refer to MIOVF register description for handling of boundary conditions.<br>Reset type: SYSRSn<br>0h (R/W) = This bit always reads back 0 and writes of 0 have no effect<br>1h (R/W) = Write a 1 to clear the task 1 interrupt overflow flag |



Copyright © 2024 Texas Instruments Incorporated

### 6.8.2.15 MIER Register (Offset = 25h) [Reset = 0000h]

MIER is shown in Figure 6-16 and described in Table 6-38.

Return to the Summary Table.

Setting the bits in the interrupt enable register (MIER) allow an incoming interrupt or main CPU software to start the corresponding CLA task. Writing a 0 will block the task, but the interrupt request will still be latched in the flag register (MIFLG). Setting the MIER register bit to 0 while the corresponding task is executing will have no effect on the task. The task will continue to run until it hits the MSTOP instruction. When a soft reset is issued, the MIER bits are cleared. There should always be at least a 1 SYSCLKOUT delay between issuing the soft reset and reconfiguring the MIER bits.

**Figure 6-16. MIER Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INT8 | INT7 | INT6 | INT5 | INT4 | INT3 | INT2 | INT1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 6-38. MIER Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R | 0h | Reserved |
| 7 | INT8 | R/W | 0h | Setting any of the bits to '1' enables the corresponding interrupt from triggering a corresponding CLA task. Writing a '0' blocks the interrupt, but the interrupt can still be latched by the MIFR register. When an interrupt is enabled and the corresponding MIFR bit is set to '1', the CLA will start executing the corresponding task and automatically clear the corresponding MIFR bit.<br>Interrupts are be serviced in normal priority order.<br>Notes: [1] If a task is currently executing and the corresponding MIER bit is cleared to '0', it will have no effect on the task. The task will run until it hits the STOP instruction.<br>Reset type: SYSRSn<br>0h (R/W) = TASK_INT_DISABLE<br>Task 8 interrupt is disabled (default)<br>1h (R/W) = TASK_INT_ENABLE<br>Task 8 interrupt is enabled |
| 6 | INT7 | R/W | 0h | Setting any of the bits to '1' enables the corresponding interrupt from triggering a corresponding CLA task. Writing a '0' blocks the interrupt, but the interrupt can still be latched by the MIFR register. When an interrupt is enabled and the corresponding MIFR bit is set to '1', the CLA will start executing the corresponding task and automatically clear the corresponding MIFR bit.<br>Interrupts are be serviced in normal priority order.<br>Notes: [1] If a task is currently executing and the corresponding MIER bit is cleared to '0', it will have no effect on the task. The task will run until it hits the STOP instruction.<br>Reset type: SYSRSn<br>0h (R/W) = TASK_INT_DISABLE<br>Task 7 interrupt is disabled (default)<br>1h (R/W) = TASK_INT_ENABLE<br>Task 7 interrupt is enabled |



Copyright © 2024 Texas Instruments Incorporated

**Table 6-38. MIER Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 5 | INT6 | R/W | 0h | Setting any of the bits to '1' enables the corresponding interrupt from triggering a corresponding CLA task. Writing a '0' blocks the interrupt, but the interrupt can still be latched by the MIFR register. When an interrupt is enabled and the corresponding MIFR bit is set to '1', the CLA will start executing the corresponding task and automatically clear the corresponding MIFR bit.<br>Interrupts are be serviced in normal priority order.<br>Notes: [1] If a task is currently executing and the corresponding MIER bit is cleared to '0', it will have no effect on the task. The task will run until it hits the STOP instruction.<br>Reset type: SYSRSn<br>0h (R/W) = TASK_INT_DISABLE<br>Task 6 interrupt is disabled (default)<br>1h (R/W) = TASK_INT_ENABLE<br>Task 6 interrupt is enabled |
| 4 | INT5 | R/W | 0h | Setting any of the bits to '1' enables the corresponding interrupt from triggering a corresponding CLA task. Writing a '0' blocks the interrupt, but the interrupt can still be latched by the MIFR register. When an interrupt is enabled and the corresponding MIFR bit is set to '1', the CLA will start executing the corresponding task and automatically clear the corresponding MIFR bit.<br>Interrupts are be serviced in normal priority order.<br>Notes: [1] If a task is currently executing and the corresponding MIER bit is cleared to '0', it will have no effect on the task. The task will run until it hits the STOP instruction.<br>Reset type: SYSRSn<br>0h (R/W) = TASK_INT_DISABLE<br>Task 5 interrupt is disabled (default)<br>1h (R/W) = TASK_INT_ENABLE<br>Task 5 interrupt is enabled |
| 3 | INT4 | R/W | 0h | Setting any of the bits to '1' enables the corresponding interrupt from triggering a corresponding CLA task. Writing a '0' blocks the interrupt, but the interrupt can still be latched by the MIFR register. When an interrupt is enabled and the corresponding MIFR bit is set to '1', the CLA will start executing the corresponding task and automatically clear the corresponding MIFR bit.<br>Interrupts are be serviced in normal priority order.<br>Notes: [1] If a task is currently executing and the corresponding MIER bit is cleared to '0', it will have no effect on the task. The task will run until it hits the STOP instruction.<br>Reset type: SYSRSn<br>0h (R/W) = TASK_INT_DISABLE<br>Task 4 interrupt is disabled (default)<br>1h (R/W) = TASK_INT_ENABLE<br>Task 4 interrupt is enabled |
| 2 | INT3 | R/W | 0h | Setting any of the bits to '1' enables the corresponding interrupt from triggering a corresponding CLA task. Writing a '0' blocks the interrupt, but the interrupt can still be latched by the MIFR register. When an interrupt is enabled and the corresponding MIFR bit is set to '1', the CLA will start executing the corresponding task and automatically clear the corresponding MIFR bit.<br>Interrupts are be serviced in normal priority order.<br>Notes: [1] If a task is currently executing and the corresponding MIER bit is cleared to '0', it will have no effect on the task. The task will run until it hits the STOP instruction.<br>Reset type: SYSRSn<br>0h (R/W) = TASK_INT_DISABLE<br>Task 3 interrupt is disabled (default)<br>1h (R/W) = TASK_INT_ENABLE<br>Task 3 interrupt is enabled |



Copyright © 2024 Texas Instruments Incorporated

**Table 6-38. MIER Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1 | INT2 | R/W | 0h | Setting any of the bits to '1' enables the corresponding interrupt from triggering a corresponding CLA task. Writing a '0' blocks the interrupt, but the interrupt can still be latched by the MIFR register. When an interrupt is enabled and the corresponding MIFR bit is set to '1', the CLA will start executing the corresponding task and automatically clear the corresponding MIFR bit.<br>Interrupts are be serviced in normal priority order.<br>Notes: [1] If a task is currently executing and the corresponding MIER bit is cleared to '0', it will have no effect on the task. The task will run until it hits the STOP instruction.<br>Reset type: SYSRSn<br>0h (R/W) = TASK_INT_DISABLE<br>Task 2 interrupt is disabled (default)<br>1h (R/W) = TASK_INT_ENABLE<br>Task 2 interrupt is enabled |
| 0 | INT1 | R/W | 0h | Setting any of the bits to '1' enables the corresponding interrupt from triggering a corresponding CLA task. Writing a '0' blocks the interrupt, but the interrupt can still be latched by the MIFR register. When an interrupt is enabled and the corresponding MIFR bit is set to '1', the CLA will start executing the corresponding task and automatically clear the corresponding MIFR bit.<br>Interrupts are be serviced in normal priority order.<br>Notes: [1] If a task is currently executing and the corresponding MIER bit is cleared to '0', it will have no effect on the task. The task will run until it hits the STOP instruction.<br>Reset type: SYSRSn<br>0h (R/W) = TASK_INT_DISABLE<br>Task 1 interrupt is disabled (default)<br>1h (R/W) = TASK_INT_ENABLE<br>Task 1 interrupt is enabled |



Copyright © 2024 Texas Instruments Incorporated

### 6.8.2.16 MIRUN Register (Offset = 26h) [Reset = 0000h]

MIRUN is shown in Figure 6-17 and described in Table 6-39.

Return to the Summary Table.

The interrupt run status register (MIRUN) indicates which task is currently executing. Only one MIRUN bit will ever be set to a 1 at any given time. The bit is automatically cleared when the task competes and the respective interrupt is fed to the peripheral interrupt expansion (PIE) block of the device. This lets the main CPU know when a task has completed. The main CPU can stop a currently running task by writing to the MCTL[SOFTRESET] bit. This will clear the MIRUN flag and stop the task. In this case no interrupt will be generated to the PIE.

**Figure 6-17. MIRUN Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INT8 | INT7 | INT6 | INT5 | INT4 | INT3 | INT2 | INT1 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 6-39. MIRUN Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R | 0h | Reserved |
| 7 | INT8 | R | 0h | These bits indicate which task is currently active. Only one bit can be set to '1' at any one time. The bit is automatically cleared to '0' when the task completes and the respective CLAINTxn line is toggled to indicate task completion. The CLAINTxn interrupt line can be fed to the PIE of the CPU so the CPU knows when a task has completed. A currently running task can be stopped by a SOFTRESET. The RUN flag is cleared, the task is stopped, but no CLAINTxn interrupt is generated.<br>Reset type: SYSRSn<br>0h (R/W) = Task 8 is not executing (default)<br>1h (R/W) = Task 8 is executing |
| 6 | INT7 | R | 0h | These bits indicate which task is currently active. Only one bit can be set to '1' at any one time. The bit is automatically cleared to '0' when the task completes and the respective CLAINTxn line is toggled to indicate task completion. The CLAINTxn interrupt line can be fed to the PIE of the CPU so the CPU knows when a task has completed. A currently running task can be stopped by a SOFTRESET. The RUN flag is cleared, the task is stopped, but no CLAINTxn interrupt is generated.<br>Reset type: SYSRSn<br>0h (R/W) = Task 7 is not executing (default)<br>1h (R/W) = Task 7 is executing |
| 5 | INT6 | R | 0h | These bits indicate which task is currently active. Only one bit can be set to '1' at any one time. The bit is automatically cleared to '0' when the task completes and the respective CLAINTxn line is toggled to indicate task completion. The CLAINTxn interrupt line can be fed to the PIE of the CPU so the CPU knows when a task has completed. A currently running task can be stopped by a SOFTRESET. The RUN flag is cleared, the task is stopped, but no CLAINTxn interrupt is generated.<br>Reset type: SYSRSn<br>0h (R/W) = Task 6 is not executing (default)<br>1h (R/W) = Task 6 is executing |



Copyright © 2024 Texas Instruments Incorporated

**Table 6-39. MIRUN Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 4 | INT5 | R | 0h | These bits indicate which task is currently active. Only one bit can be set to '1' at any one time. The bit is automatically cleared to '0' when the task completes and the respective CLAINTxn line is toggled to indicate task completion. The CLAINTxn interrupt line can be fed to the PIE of the CPU so the CPU knows when a task has completed. A currently running task can be stopped by a SOFTRESET. The RUN flag is cleared, the task is stopped, but no CLAINTxn interrupt is generated.<br>Reset type: SYSRSn<br>0h (R/W) = Task 5 is not executing (default)<br>1h (R/W) = Task 5 is executing |
| 3 | INT4 | R | 0h | These bits indicate which task is currently active. Only one bit can be set to '1' at any one time. The bit is automatically cleared to '0' when the task completes and the respective CLAINTxn line is toggled to indicate task completion. The CLAINTxn interrupt line can be fed to the PIE of the CPU so the CPU knows when a task has completed. A currently running task can be stopped by a SOFTRESET. The RUN flag is cleared, the task is stopped, but no CLAINTxn interrupt is generated.<br>Reset type: SYSRSn<br>0h (R/W) = Task 4 is not executing (default)<br>1h (R/W) = Task 4 is executing |
| 2 | INT3 | R | 0h | These bits indicate which task is currently active. Only one bit can be set to '1' at any one time. The bit is automatically cleared to '0' when the task completes and the respective CLAINTxn line is toggled to indicate task completion. The CLAINTxn interrupt line can be fed to the PIE of the CPU so the CPU knows when a task has completed. A currently running task can be stopped by a SOFTRESET. The RUN flag is cleared, the task is stopped, but no CLAINTxn interrupt is generated.<br>Reset type: SYSRSn<br>0h (R/W) = Task 3 is not executing (default)<br>1h (R/W) = Task 3 is executing |
| 1 | INT2 | R | 0h | These bits indicate which task is currently active. Only one bit can be set to '1' at any one time. The bit is automatically cleared to '0' when the task completes and the respective CLAINTxn line is toggled to indicate task completion. The CLAINTxn interrupt line can be fed to the PIE of the CPU so the CPU knows when a task has completed. A currently running task can be stopped by a SOFTRESET. The RUN flag is cleared, the task is stopped, but no CLAINTxn interrupt is generated.<br>Reset type: SYSRSn<br>0h (R/W) = Task 2 is not executing (default)<br>1h (R/W) = Task 2 is executing |
| 0 | INT1 | R | 0h | These bits indicate which task is currently active. Only one bit can be set to '1' at any one time. The bit is automatically cleared to '0' when the task completes and the respective CLAINTxn line is toggled to indicate task completion. The CLAINTxn interrupt line can be fed to the PIE of the CPU so the CPU knows when a task has completed. A currently running task can be stopped by a SOFTRESET. The RUN flag is cleared, the task is stopped, but no CLAINTxn interrupt is generated.<br>Reset type: SYSRSn<br>0h (R/W) = Task 1 is not executing (default)<br>1h (R/W) = Task 1 is executing |



Copyright © 2024 Texas Instruments Incorporated

### 6.8.2.17 _MPC Register (Offset = 28h) [Reset = 0000h]

_MPC is shown in Figure 6-18 and described in Table 6-40.

Return to the Summary Table.

CLA Program Counter

**Figure 6-18. _MPC Register**

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

**Table 6-40. _MPC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | _MPC | R | 0h | Program Counter: The PC value is initialized by the appropriate MVECTx register when an interrupt (task) is serviced.<br>The MPC register address 16-bits and not 32-bits. Hence the address range of the CLA with a 16-bit MPC is 64Kx16 words or 32K CLA instructions.<br>Notes: [1] To be consistent with C28 core implementation, the PC value points to the instruction in D2 stage of pipeline.<br>[2] After a STOP operation, and with no other task pending, the PC will remain pointing to the STOP operation.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 6.8.2.18 _MAR0 Register (Offset = 2Ah) [Reset = 0000h]

_MAR0 is shown in Figure 6-19 and described in Table 6-41.

Return to the Summary Table.

CLA Auxiliary Register 0

**Figure 6-19. _MAR0 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| _MAR0 | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| _MAR0 | | | | | | | |
| R-0h | | | | | | | |

**Table 6-41. _MAR0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | _MAR0 | R | 0h | CLA Auxillary Register 0<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 6.8.2.19 _MAR1 Register (Offset = 2Bh) [Reset = 0000h]

_MAR1 is shown in Figure 6-20 and described in Table 6-42.

Return to the Summary Table.

CLA Auxiliary Register 1

**Figure 6-20. _MAR1 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| _MAR1 | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| _MAR1 | | | | | | | |
| R-0h | | | | | | | |

**Table 6-42. _MAR1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | _MAR1 | R | 0h | CLA Auxillary Register 1<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 6.8.2.20 _MSTF Register (Offset = 2Eh) [Reset = 00000000h]

_MSTF is shown in Figure 6-21 and described in Table 6-43.

Return to the Summary Table.

The CLA status register (MSTF) reflects the results of different operations. These are the basic rules for the flags:
- Zero and negative flags are cleared or set based on:
- floating-point moves to registers
- the result of compare, minimum, maximum, negative and absolute value operations
- the integer result of operations such as MMOV16, MAND32, MOR32, MXOR32, MCMP32, MASR32, MLSR32
- Overflow and underflow flags are set by floating-point math instructions such as multiply, add, subtract and 1/x. These flags may also be connected to the peripheral interrupt expansion (PIE) block on your device. This can be useful for debugging underflow and overflow conditions within an application.

**Figure 6-21. _MSTF Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | _RPC | | | |
| R-0h | | | | R-0h | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| _RPC | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| _RPC | | | | MEALLOW | RESERVED | RNDF32 | RESERVED |
| R-0h | | | | R-0h | R-0h | R-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | TF | RESERVED | | ZF | NF | LUF | LVF |
| R-0h | R-0h | R-0h | | R-0h | R-0h | R-0h | R-0h |

**Table 6-43. _MSTF Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-28 | RESERVED | R | 0h | Reserved |
| 27-12 | _RPC | R | 0h | Return program counter<br>The _RPC is used to save and restore the MPC address by the MCCNDD and MRCNDD operations<br>Reset type: SYSRSn |
| 11 | MEALLOW | R | 0h | MEALLOW Status<br>This bit enables and disables CLA write access to EALLOW protected registers This is independent of the state of the EALLOW bit in the main CPU status register This status bit can be saved and restored by the MMOV32 STF, mem32 instruction<br>Reset type: SYSRSn<br>0h (R/W) = The CLA cannot write to EALLOW protected registers. This bit is cleared by the CLA instruction, MEDIS.<br>1h (R/W) = The CLA is allowed to write to EALLOW protected registers. This bit is set by the CLA instruction, MEALLOW. |
| 10 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 6-43. _MSTF Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 9 | RNDF32 | R | 0h | Round 32-bit Floating-Point Mode<br>Use the MSETFLG and MMOV32 MSTF, mem32 instructions to change the rounding mode<br>Reset type: SYSRSn<br>0h (R/W) = If this bit is zero, the MMPYF32, MADDF32 and MSUBF32 instructions will round to zero (truncate).<br>1h (R/W) = If this bit is one, the MMPYF32, MADDF32 and MSUBF32 instructions will round to the nearest even value. |
| 8-7 | RESERVED | R | 0h | Reserved |
| 6 | TF | R | 0h | Test Flag<br>The MTESTTF instruction can modify this flag based on the condition tested The MSETFLG and MMOV32 MSTF, mem32 instructions can also be used to modify this flag<br>Reset type: SYSRSn<br>0h (R/W) = The condition tested with the MTESTTF instruction is false.<br>1h (R/W) = The condition tested with the MTESTTF instruction is true. |
| 5-4 | RESERVED | R | 0h | Reserved |
| 3 | ZF | R | 0h | Zero Flag<br>- Instructions that modify this flag based on the floating-point value stored in the destination register:<br>MMOV32, MMOVD32, MABSF32, MNEGF32<br>- Instructions that modify this flag based on the floating-point result of the operation:<br>MCMPF32, MMAXF32, and MMINF32<br>- Instructions that modify this flag based on the integer result of the operation:<br>MMOV16, MAND32, MOR32, MXOR32, MCMP32, MASR32, MLSR32 and<br>MLSL32<br>The MSETFLG and MMOV32 MSTF, mem32 instructions can also be used to modify this flag<br>Reset type: SYSRSn<br>0h (R/W) = The value is not zero<br>1h (R/W) = The value is zero |
| 2 | NF | R | 0h | Negative Flag<br>- Instructions that modify this flag based on the floating-point value stored in the destination register:<br>MMOV32, MMOVD32, MABSF32, MNEGF32<br>- Instructions that modify this flag based on the floating-point result of the operation:<br>MCMPF32, MMAXF32, and MMINF32<br>- Instructions that modify this flag based on the integer result of the operation:<br>MMOV16, MAND32, MOR32, MXOR32, MCMP32, MASR32, MLSR32 and<br>MLSL32<br>The MSETFLG and MMOV32 MSTF, mem32 instructions can also be used to modify this flag<br>Reset type: SYSRSn<br>0h (R/W) = The value is not negative<br>1h (R/W) = The value is negative |



Copyright © 2024 Texas Instruments Incorporated

**Table 6-43. _MSTF Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1 | LUF | R | 0h | Latched Underflow Flag<br>The following instructions will set this flag to 1 if an underflow occurs: MMPYF32, MADDF32,<br>MSUBF32, MMACF32, MEINVF32, MEISQRTF32<br>The MSETFLG and MMOV32 MSTF, mem32 instructions can also be used to modify this flag<br>Reset type: SYSRSn<br>0h (R/W) = An underflow condition has not been latched<br>1h (R/W) = An underflow condition has been latched |
| 0 | LVF | R | 0h | Latched Overflow Flag<br>The following instructions will set this flag to 1 if an overflow occurs: MMPYF32, MADDF32, MSUBF32, MMACF32, MEINVF32, MEISQRTF32<br>The MSETFLG and MMOV32 MSTF, mem32 instructions can also be used to modify this flag<br>Reset type: SYSRSn<br>0h (R/W) = An overflow condition has not been latched<br>1h (R/W) = An overflow condition has been latched |



Copyright © 2024 Texas Instruments Incorporated

### 6.8.2.21 _MR0 Register (Offset = 30h) [Reset = 00000000h]

_MR0 is shown in Figure 6-22 and described in Table 6-44.

Return to the Summary Table.

CLA Floating-Point Result Register 0

**Figure 6-22. _MR0 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| i32 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 6-44. _MR0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | i32 | R | 0h | CLA Result Register 0<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 6.8.2.22 _MR1 Register (Offset = 34h) [Reset = 00000000h]

_MR1 is shown in Figure 6-23 and described in Table 6-45.

Return to the Summary Table.

CLA Floating-Point Result Register 1

**Figure 6-23. _MR1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| i32 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 6-45. _MR1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | i32 | R | 0h | CLA Result Register 1<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 6.8.2.23 _MR2 Register (Offset = 38h) [Reset = 00000000h]

_MR2 is shown in Figure 6-24 and described in Table 6-46.

Return to the Summary Table.

CLA Floating-Point Result Register 2

**Figure 6-24. _MR2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| i32 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 6-46. _MR2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | i32 | R | 0h | CLA Result Register 2<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 6.8.2.24 _MR3 Register (Offset = 3Ch) [Reset = 00000000h]

_MR3 is shown in Figure 6-25 and described in Table 6-47.

Return to the Summary Table.

CLA Floating-Point Result Register 3

**Figure 6-25. _MR3 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| i32 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 6-47. _MR3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | i32 | R | 0h | CLA Result Register 3<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 6.8.3 CLA_SOFTINT_REGS Registers

Table 6-48 lists the memory-mapped registers for the CLA_SOFTINT_REGS registers. All register offset addresses not listed in Table 6-48 should be considered as reserved locations and the register contents should not be modified.

**Table 6-48. CLA_SOFTINT_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | SOFTINTEN | CLA Software Interrupt Enable Register | | Go |
| 2h | SOFTINTFRC | CLA Software Interrupt Force Register | | Go |

Complex bit access types are encoded to fit into small table cells. Table 6-49 shows the codes that are used for access types in this section.

**Table 6-49. CLA_SOFTINT_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| R-0 | R<br>-0 | Read<br>Returns 0s |
| Write Type | | |
| W | W | Write |
| W1S | W<br>1S | Write<br>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. |



Copyright © 2024 Texas Instruments Incorporated

### 6.8.3.1 SOFTINTEN Register (Offset = 0h) [Reset = 0000h]

SOFTINTEN is shown in Figure 6-26 and described in Table 6-50.

Return to the Summary Table.

Enables the ability to generate CLA task interrupt from within the task, by writing to SOFTINTFRC register. SOFTINTFRC register can only be written from CLA.

**Figure 6-26. SOFTINTEN Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R/W-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TASK8 | TASK7 | TASK6 | TASK5 | TASK4 | TASK3 | TASK2 | TASK1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 6-50. SOFTINTEN Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R/W | 0h | Reserved |
| 7 | TASK8 | R/W | 0h | 0: End-Of-Task Interrupt is fired for the respective task<br>1: Enable Software Interrupt for the respective task. End-of-Task interrupt is not sent to CPU in this case.<br>Note: SOFTINTEN register is read only in the CPU memory map.<br>Reset type: SYSRSn |
| 6 | TASK7 | R/W | 0h | 0: End-Of-Task Interrupt is fired for the respective task<br>1: Enable Software Interrupt for the respective task. End-of-Task interrupt is not sent to CPU in this case.<br>Note: SOFTINTEN register is read only in the CPU memory map.<br>Reset type: SYSRSn |
| 5 | TASK6 | R/W | 0h | 0: End-Of-Task Interrupt is fired for the respective task<br>1: Enable Software Interrupt for the respective task. End-of-Task interrupt is not sent to CPU in this case.<br>Note: SOFTINTEN register is read only in the CPU memory map.<br>Reset type: SYSRSn |
| 4 | TASK5 | R/W | 0h | 0: End-Of-Task Interrupt is fired for the respective task<br>1: Enable Software Interrupt for the respective task. End-of-Task interrupt is not sent to CPU in this case.<br>Note: SOFTINTEN register is read only in the CPU memory map.<br>Reset type: SYSRSn |
| 3 | TASK4 | R/W | 0h | 0: End-Of-Task Interrupt is fired for the respective task<br>1: Enable Software Interrupt for the respective task. End-of-Task interrupt is not sent to CPU in this case.<br>Note: SOFTINTEN register is read only in the CPU memory map.<br>Reset type: SYSRSn |
| 2 | TASK3 | R/W | 0h | 0: End-Of-Task Interrupt is fired for the respective task<br>1: Enable Software Interrupt for the respective task. End-of-Task interrupt is not sent to CPU in this case.<br>Note: SOFTINTEN register is read only in the CPU memory map.<br>Reset type: SYSRSn |
| 1 | TASK2 | R/W | 0h | 0: End-Of-Task Interrupt is fired for the respective task<br>1: Enable Software Interrupt for the respective task. End-of-Task interrupt is not sent to CPU in this case.<br>Note: SOFTINTEN register is read only in the CPU memory map.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 6-50. SOFTINTEN Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | TASK1 | R/W | 0h | 0: End-Of-Task Interrupt is fired for the respective task<br>1: Enable Software Interrupt for the respective task. End-of-Task interrupt is not sent to CPU in this case.<br>Note: SOFTINTEN register is read only in the CPU memory map.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 6.8.3.2 SOFTINTFRC Register (Offset = 2h) [Reset = 0000h]

SOFTINTFRC is shown in Figure 6-27 and described in Table 6-51.

Return to the Summary Table.

Writing a value of 1 in a bit will generate the corresponding task interrupt.This register is only accessible by the CLA (not the CPU).

**Figure 6-27. SOFTINTFRC Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R/W-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TASK8 | TASK7 | TASK6 | TASK5 | TASK4 | TASK3 | TASK2 | TASK1 |
| R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h |

**Table 6-51. SOFTINTFRC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R/W | 0h | Reserved |
| 7 | TASK8 | R-0/W1S | 0h | Write of '1' will generate a CLA software interrupt to the CPU for the corresponding task.<br>Write of '0' has no effect.<br>Reset type: SYSRSn |
| 6 | TASK7 | R-0/W1S | 0h | Write of '1' will generate a CLA software interrupt to the CPU for the corresponding task.<br>Write of '0' has no effect.<br>Reset type: SYSRSn |
| 5 | TASK6 | R-0/W1S | 0h | Write of '1' will generate a CLA software interrupt to the CPU for the corresponding task.<br>Write of '0' has no effect.<br>Reset type: SYSRSn |
| 4 | TASK5 | R-0/W1S | 0h | Write of '1' will generate a CLA software interrupt to the CPU for the corresponding task.<br>Write of '0' has no effect.<br>Reset type: SYSRSn |
| 3 | TASK4 | R-0/W1S | 0h | Write of '1' will generate a CLA software interrupt to the CPU for the corresponding task.<br>Write of '0' has no effect.<br>Reset type: SYSRSn |
| 2 | TASK3 | R-0/W1S | 0h | Write of '1' will generate a CLA software interrupt to the CPU for the corresponding task.<br>Write of '0' has no effect.<br>Reset type: SYSRSn |
| 1 | TASK2 | R-0/W1S | 0h | Write of '1' will generate a CLA software interrupt to the CPU for the corresponding task.<br>Write of '0' has no effect.<br>Reset type: SYSRSn |
| 0 | TASK1 | R-0/W1S | 0h | Write of '1' will generate a CLA software interrupt to the CPU for the corresponding task.<br>Write of '0' has no effect.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 6.8.4 CLA Registers to Driverlib Functions

**Table 6-52. CLA Registers to Driverlib Functions**

| File | Driverlib Function |
|---|---|
| **MVECT1** | |
| cla.h | CLA_mapTaskVector |
| **MVECT2** | |
| - | See MVECT1 |
| **MVECT3** | |
| - | See MVECT1 |
| **MVECT4** | |
| - | See MVECT1 |
| **MVECT5** | |
| - | See MVECT1 |
| **MVECT6** | |
| - | See MVECT1 |
| **MVECT7** | |
| - | See MVECT1 |
| **MVECT8** | |
| - | See MVECT1 |
| **MCTL** | |
| cla.h | CLA_performHardReset |
| cla.h | CLA_performSoftReset |
| cla.h | CLA_enableIACK |
| cla.h | CLA_disableIACK |
| **MIFR** | |
| cla.h | CLA_getPendingTaskFlag |
| cla.h | CLA_getAllPendingTaskFlags |
| cla.h | CLA_forceTasks |
| **MIOVF** | |
| cla.h | CLA_getTaskOverflowFlag |
| cla.h | CLA_getAllTaskOverflowFlags |
| **MIFRC** | |
| cla.h | CLA_forceTasks |
| **MICLR** | |
| cla.h | CLA_clearTaskFlags |
| **MICLROVF** | |
| - | |
| **MIER** | |
| cla.h | CLA_enableTasks |
| cla.h | CLA_disableTasks |
| **MIRUN** | |
| cla.h | CLA_getTaskRunStatus |
| cla.h | CLA_getAllTaskRunStatus |
| **MPC** | |
| - | |
| **MAR0** | |



Copyright © 2024 Texas Instruments Incorporated

**Table 6-52. CLA Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| - | |
| **MAR1** | |
| - | |
| **MSTF** | |
| - | |
| **MR0** | |
| - | |
| **MR1** | |
| - | |
| **MR2** | |
| - | |
| **MR3** | |
| - | |
| **SOFTINTEN** | |
| cla.h | CLA_enableSoftwareInterrupt |
| cla.h | CLA_disableSoftwareInterrupt |
| **SOFTINTFRC** | |
| cla.h | CLA_forceSoftwareInterrupt |



Copyright © 2024 Texas Instruments Incorporated

Chapter 7

# Interprocessor Communication (IPC)

The Interprocessor Communications (IPC) module allows communication between the two CPU subsystems.


**7.1 Introduction** ..... 917
**7.2 Message RAMs** ..... 918
**7.3 IPC Flags and Interrupts** ..... 918
**7.4 IPC Command Registers** ..... 918
**7.5 Free-Running Counter** ..... 918
**7.6 IPC Communication Protocol** ..... 919
**7.7 IPC Registers** ..... 920





Copyright © 2024 Texas Instruments Incorporated

## 7.1 Introduction

This section details the IPC features that each CPU can use to request and share information. The IPC features are:

- Message RAMs
- IPC flags and interrupts
- IPC command registers
- Flash pump semaphore
- Clock configuration semaphore
- Free-running counter

All IPC features are independent of each other, and most do not require any specific data format.

There are also two registers for boot mode and status communication. Please refer to the boot ROM chapter for more information on these registers.

Figure 7-1 shows the design structure of the IPC module.

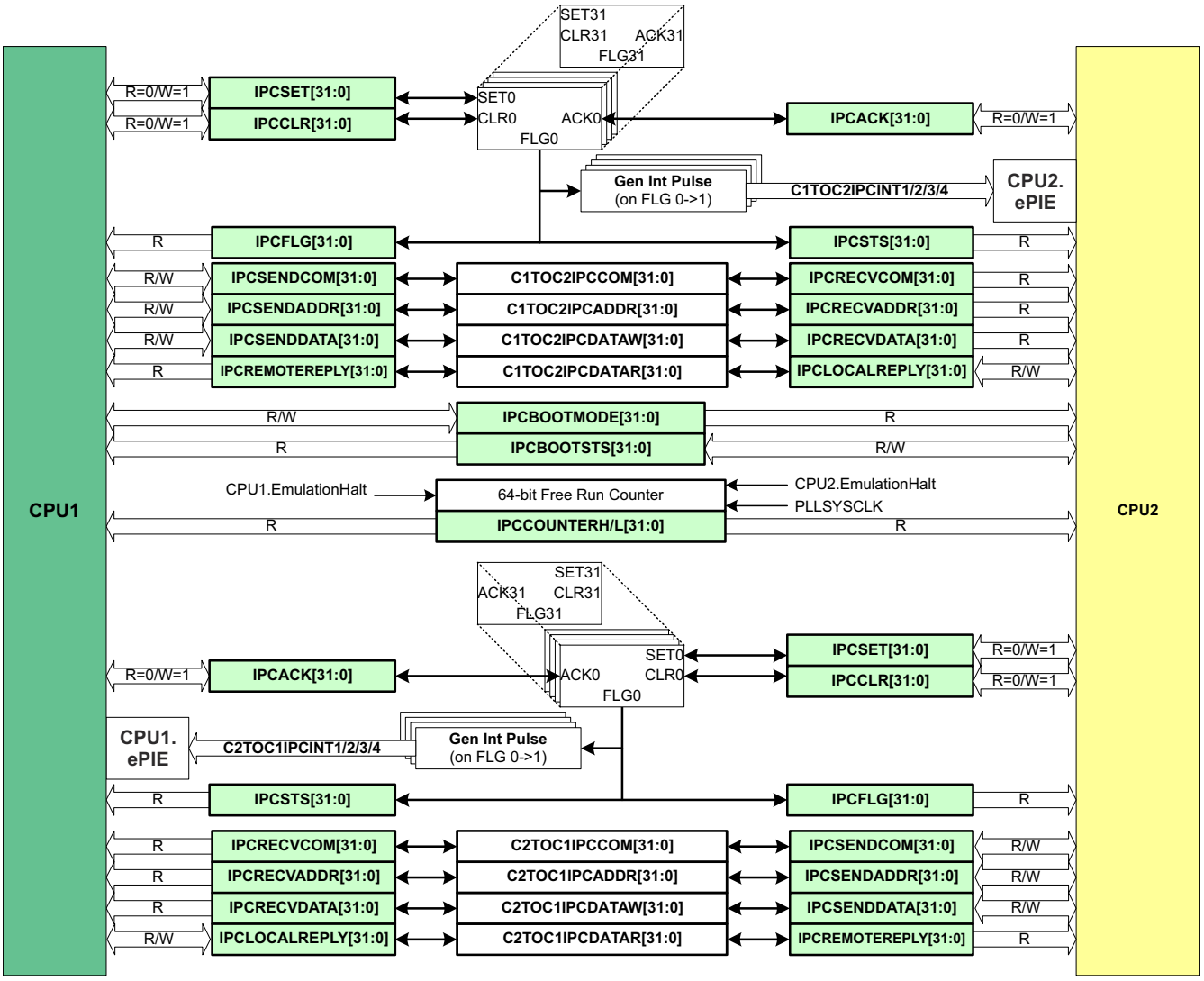



**Figure 7-1. IPC Module Architecture**



Copyright © 2024 Texas Instruments Incorporated

## 7.2 Message RAMs

There are two dedicated 2-kB blocks of message RAM. Each CPU and the DMA have read and write access to one RAM and read-only access to the other RAM, as shown in Table 7-1..

Reading or writing a message RAM does not trigger any events on the remote CPU.

**Table 7-1. IPC Message RAM Read/Write Access**

| | CPU1 | CPU2 | CPU1 DMA | CPU2 DMA |
|---|---|---|---|---|
| CPU1 to CPU2 (1K x 16, address 0x03FC00) | R/W | R | R/W | R |
| CPU2 to CPU1 (1K x 16, address 0x03F800) | R | R/W | R | R/W |

## 7.3 IPC Flags and Interrupts

There are 32 IPC event signals in each direction between the CPU pairs. These signals can be used for flag-based event polling. With the C28x core, four of them (IPC0 - IPC3) can be configured to generate IPC interrupts on the remote CPU.

## 7.4 IPC Command Registers

The IPC command registers provide a simple and flexible way for the CPUs to exchange more complex messages. Each CPU has eight dedicated registers; four for sending messages and four for receiving messages. The register names were chosen to support a simple command/response protocol, but can be used for any purpose. Only the read/write permissions are determined by hardware; the data format is entirely software-defined.

For sending messages, each CPU has three writable registers and one read-only register. Those same registers are accessible on the remote CPU as three read-only registers and one writable register. Table 7-2 shows the command registers.

**Table 7-2. IPC Command Registers**

| Local Register Name | Local CPU | Remote CPU | Remote Register Name |
|---|---|---|---|
| IPCSENDCOM | R/W | R | IPCRECVCOM |
| IPCSENDADDR | R/W | R | IPCRECVADDR |
| IPCSENDDATA | R/W | R | IPCRECVDATA |
| IPCREMOTEREPLY | R | R/W | IPCLOCALREPLY |

## 7.5 Free-Running Counter

A 64-bit free-running counter is present in the device and can be used to timestamp IPC events between processors. The counter is clocked by PLLSYSCLK and reset by SYSRSn. The counter is implemented as two 32-bit registers, IPCCOUNTERH and IPCCOUNTERL. When IPCCOUNTERL is read, the value of IPCCOUNTERH is saved. A subsequent read to IPCCOUNTERH returns this saved value. Therefore, the user must always read IPCCOUNTERL first then read IPCCOUNTERH next. This design prevents race conditions due to IPCCOUNTERL overflowing between reads of the two registers.

The free-running counter stops only when emulation is suspended (when debugger hits a breakpoint) on all CPUs. If any core is executing, the counter runs.



Copyright © 2024 Texas Instruments Incorporated

## 7.6 IPC Communication Protocol

This section describes the hardware support options for IPC communication between the two CPUs. These options can be used independently or in combination. All flag definitions and data formats are entirely user-defined.

- The flag system supports event-based communication via interrupts and register polling.
  - CPUx can raise an IPC event by writing to any of the 32 bits of the IPCSET register. This sets the corresponding bits in the CPUx IPCFLG register and CPUy IPCSTS register.
  - CPUy can signal the response to the event by setting the appropriate bit in the IPCACK register. This clears the corresponding bits in the CPUx IPCFLG register and the CPUy IPCSTS register.
  - If CPUx needs to cancel an event, CPUx can set the appropriate bit in the IPCCLR register. This has the same effect as CPUy writing to IPCACK.
  - Flags 0–3 (set using IPCSET[3:0]) fire interrupts to the remote CPU. The remote CPU must configure the ePIE module properly to receive an IPC interrupt. Flags 4–31 (set using IPCSET[31:4]) do not produce interrupts. Multiple flags can be set, acknowledged, and cleared simultaneously.
- The command registers support sending several distinct pieces of information and are named COM, ADDR, DATA, and REPLY for convenience only and can hold whatever data the application needs.
  - CPUx can write data to the IPCSENDCOM, IPCSENDADDR, and IPCSENDDATA registers. CPUy receives these in the IPCRECVCOM, IPCRECVADDR, and IPCRECVDATA registers.
  - CPUy can respond by writing to its IPCLOCALREPLY register. CPUx receives this data in its ownIPCREMOTEREPLY register.
- There is an additional pair of command-like registers offered for boot-time IPC or any other convenient use — IPCBOOTMODE and IPCBOOTSTS. Both CPUs can read these registers. CPUx can only write to IPCBOOTMODE, and CPUy can only write to IPCBOOTSTS.
- There are two shared memories for passing large amounts of data between the CPUs. Each CPU has a writable memory for sending data and a read-only memory for receiving data.
- Here is an example of how to use these features together. CPUx needs some data from CPUy's LS RAM. The data is at CPUy address 0x9400 and is 0x80 16-bit words long. The protocol can be implemented like this:
  - CPUx writes 0x1 to IPCSENDCOM, defined in software to mean "copy data from address". CPUx writes the address (0x9400) to IPCSENDADDR and the data length (0x80) to IPCSENDDATA.
  - CPUx writes to IPCSET[3] and IPCSET[16]. Here, IPC flag 3 is configured to send an interrupt and IPCSET[16] is defined in software to indicate an incoming command. CPUx begins polling for IPCFLG[3] to go low.
  - CPUy receives the interrupt. In the interrupt handler, CPUy checks IPCSTS, finds that flag 16 is set, and runs a command processor.
  - CPUy reads the command (0x1) from IPCRECVCOM, the address (0x9400) from IPCRECVADDR, and the data length (0x80) from IPCRECVDATA. CPUy then copies the LS RAM data to an empty space in the writable shared memory starting at offset 0x210.
  - CPUy writes the shared memory address (0x210) to the IPCLOCALREPLY register. CPUy then writes to IPCACK[16] and IPCACK[3] to clear the flags and indicate completion of the command. CPUy's work is done.
  - CPUx sees IPCFLG[3] go low. CPUx reads IPCREMOTEREPLY to get the shared memory offset of the copied data (0x210).



Copyright © 2024 Texas Instruments Incorporated

## 7.7 IPC Registers

This section describes the Interprocessor Communication Registers.

### *7.7.1 IPC Base Addresses*

**Table 7-3. IPC Base Addresses**

| Device Registers | Register Name | Start Address | End Address |
|---|---|---|---|
| IpcRegs (CPU1) | IPC_REGS_CPU1 | 0x0005_0000 | 0x0005_0023 |
| IpcRegs (CPU2) | IPC_REGS_CPU2 | 0x0005_0000 | 0x0005_0023 |



Copyright © 2024 Texas Instruments Incorporated

### 7.7.2 IPC_REGS_CPU1 Registers

Table 7-4 lists the memory-mapped registers for the IPC_REGS_CPU1 registers. All register offset addresses not listed in Table 7-4 should be considered as reserved locations and the register contents should not be modified.

**Table 7-4. IPC_REGS_CPU1 Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | IPCACK | IPC incoming flag clear (acknowledge) register | | Go |
| 2h | IPCSTS | IPC incoming flag status register | | Go |
| 4h | IPCSET | IPC remote flag set register | | Go |
| 6h | IPCCLR | IPC remote flag clear register | | Go |
| 8h | IPCFLG | IPC remote flag status register | | Go |
| Ch | IPCCOUNTERL | IPC Counter Low Register | | Go |
| Eh | IPCCOUNTERH | IPC Counter High Register | | Go |
| 10h | IPCSENDCOM | Local to Remote IPC Command Register | | Go |
| 12h | IPCSENDADDR | Local to Remote IPC Address Register | | Go |
| 14h | IPCSENDDATA | Local to Remote IPC Data Register | | Go |
| 16h | IPCREMOTEREPLY | Remote to Local IPC Reply Data Register | | Go |
| 18h | IPCRECVCOM | Remote to Local IPC Command Register | | Go |
| 1Ah | IPCRECVADDR | Remote to Local IPC Address Register | | Go |
| 1Ch | IPCRECVDATA | Remote to Local IPC Data Register | | Go |
| 1Eh | IPCLOCALREPLY | Local to Remote IPC Reply Data Register | | Go |
| 20h | IPCBOOTSTS | CPU2 to CPU1 IPC Boot Status Register | | Go |
| 22h | IPCBOOTMODE | CPU1 to CPU2 IPC Boot Mode Register | | Go |

Complex bit access types are encoded to fit into small table cells. Table 7-5 shows the codes that are used for access types in this section.

**Table 7-5. IPC_REGS_CPU1 Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| R-0 | R<br>-0 | Read<br>Returns 0s |
| Write Type | | |
| W | W | Write |
| W1S | W<br>1S | Write<br>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. |



Copyright © 2024 Texas Instruments Incorporated

### 7.7.2.1 IPCACK Register (Offset = 0h) [Reset = 00000000h]

IPCACK is shown in Figure 7-2 and described in Table 7-6.

Return to the Summary Table.

IPC incoming flag clear (acknowledge) register

**Figure 7-2. IPCACK Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| IPC31 | IPC30 | IPC29 | IPC28 | IPC27 | IPC26 | IPC25 | IPC24 |
| R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| IPC23 | IPC22 | IPC21 | IPC20 | IPC19 | IPC18 | IPC17 | IPC16 |
| R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| IPC15 | IPC14 | IPC13 | IPC12 | IPC11 | IPC10 | IPC9 | IPC8 |
| 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 |
| IPC7 | IPC6 | IPC5 | IPC4 | IPC3 | IPC2 | IPC1 | IPC0 |
| 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 7-6. IPCACK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | IPC31 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC31 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 30 | IPC30 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC30 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 29 | IPC29 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC29 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 28 | IPC28 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC28 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 27 | IPC27 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC27 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 26 | IPC26 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC26 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 25 | IPC25 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC25 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 7-6. IPCACK Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 24 | IPC24 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC24 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 23 | IPC23 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC23 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 22 | IPC22 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC22 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 21 | IPC21 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC21 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 20 | IPC20 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC20 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 19 | IPC19 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC19 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 18 | IPC18 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC18 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 17 | IPC17 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC17 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 16 | IPC16 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC16 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 15 | IPC15 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC15 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 14 | IPC14 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC14 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 13 | IPC13 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC13 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 12 | IPC12 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC12 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 11 | IPC11 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC11 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 7-6. IPCACK Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 10 | IPC10 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC10 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 9 | IPC9 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC9 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 8 | IPC8 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC8 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 7 | IPC7 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC7 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 6 | IPC6 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC6 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 5 | IPC5 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC5 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 4 | IPC4 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC4 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 3 | IPC3 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC3 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 2 | IPC2 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC2 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 1 | IPC1 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC1 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 0 | IPC0 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC0 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 7.7.2.2 IPCSTS Register (Offset = 2h) [Reset = 00000000h]

IPCSTS is shown in Figure 7-3 and described in Table 7-7.

Return to the Summary Table.

IPC incoming flag status register

**Figure 7-3. IPCSTS Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| IPC31 | IPC30 | IPC29 | IPC28 | IPC27 | IPC26 | IPC25 | IPC24 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| IPC23 | IPC22 | IPC21 | IPC20 | IPC19 | IPC18 | IPC17 | IPC16 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| IPC15 | IPC14 | IPC13 | IPC12 | IPC11 | IPC10 | IPC9 | IPC8 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| IPC7 | IPC6 | IPC5 | IPC4 | IPC3 | IPC2 | IPC1 | IPC0 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 7-7. IPCSTS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | IPC31 | R | 0h | Indicates to the local CPU if the IPC31 event flag was set by the remote CPU.<br>0: No IPC31 event was set by the remote CPU<br>1: An IPC31 event was set by the remote CPU<br>Reset type: SYSRSn |
| 30 | IPC30 | R | 0h | Indicates to the local CPU if the IPC30 event flag was set by the remote CPU.<br>0: No IPC30 event was set by the remote CPU<br>1: An IPC30 event was set by the remote CPU<br>Reset type: SYSRSn |
| 29 | IPC29 | R | 0h | Indicates to the local CPU if the IPC29 event flag was set by the remote CPU.<br>0: No IPC29 event was set by the remote CPU<br>1: An IPC29 event was set by the remote CPU<br>Reset type: SYSRSn |
| 28 | IPC28 | R | 0h | Indicates to the local CPU if the IPC28 event flag was set by the remote CPU.<br>0: No IPC28 event was set by the remote CPU<br>1: An IPC28 event was set by the remote CPU<br>Reset type: SYSRSn |
| 27 | IPC27 | R | 0h | Indicates to the local CPU if the IPC27 event flag was set by the remote CPU.<br>0: No IPC27 event was set by the remote CPU<br>1: An IPC27 event was set by the remote CPU<br>Reset type: SYSRSn |
| 26 | IPC26 | R | 0h | Indicates to the local CPU if the IPC26 event flag was set by the remote CPU.<br>0: No IPC26 event was set by the remote CPU<br>1: An IPC26 event was set by the remote CPU<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 7-7. IPCSTS Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 25 | IPC25 | R | 0h | Indicates to the local CPU if the IPC25 event flag was set by the remote CPU.<br>0: No IPC25 event was set by the remote CPU<br>1: An IPC25 event was set by the remote CPU<br>Reset type: SYSRSn |
| 24 | IPC24 | R | 0h | Indicates to the local CPU if the IPC24 event flag was set by the remote CPU.<br>0: No IPC24 event was set by the remote CPU<br>1: An IPC24 event was set by the remote CPU<br>Reset type: SYSRSn |
| 23 | IPC23 | R | 0h | Indicates to the local CPU if the IPC23 event flag was set by the remote CPU.<br>0: No IPC23 event was set by the remote CPU<br>1: An IPC23 event was set by the remote CPU<br>Reset type: SYSRSn |
| 22 | IPC22 | R | 0h | Indicates to the local CPU if the IPC22 event flag was set by the remote CPU.<br>0: No IPC22 event was set by the remote CPU<br>1: An IPC22 event was set by the remote CPU<br>Reset type: SYSRSn |
| 21 | IPC21 | R | 0h | Indicates to the local CPU if the IPC21 event flag was set by the remote CPU.<br>0: No IPC21 event was set by the remote CPU<br>1: An IPC21 event was set by the remote CPU<br>Reset type: SYSRSn |
| 20 | IPC20 | R | 0h | Indicates to the local CPU if the IPC20 event flag was set by the remote CPU.<br>0: No IPC20 event was set by the remote CPU<br>1: An IPC20 event was set by the remote CPU<br>Reset type: SYSRSn |
| 19 | IPC19 | R | 0h | Indicates to the local CPU if the IPC19 event flag was set by the remote CPU.<br>0: No IPC19 event was set by the remote CPU<br>1: An IPC19 event was set by the remote CPU<br>Reset type: SYSRSn |
| 18 | IPC18 | R | 0h | Indicates to the local CPU if the IPC18 event flag was set by the remote CPU.<br>0: No IPC18 event was set by the remote CPU<br>1: An IPC18 event was set by the remote CPU<br>Reset type: SYSRSn |
| 17 | IPC17 | R | 0h | Indicates to the local CPU if the IPC17 event flag was set by the remote CPU.<br>0: No IPC17 event was set by the remote CPU<br>1: An IPC17 event was set by the remote CPU<br>Reset type: SYSRSn |
| 16 | IPC16 | R | 0h | Indicates to the local CPU if the IPC16 event flag was set by the remote CPU.<br>0: No IPC16 event was set by the remote CPU<br>1: An IPC16 event was set by the remote CPU<br>Reset type: SYSRSn |
| 15 | IPC15 | R | 0h | Indicates to the local CPU if the IPC15 event flag was set by the remote CPU.<br>0: No IPC15 event was set by the remote CPU<br>1: An IPC15 event was set by the remote CPU<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 7-7. IPCSTS Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 14 | IPC14 | R | 0h | Indicates to the local CPU if the IPC14 event flag was set by the remote CPU.<br>0: No IPC14 event was set by the remote CPU<br>1: An IPC14 event was set by the remote CPU<br>Reset type: SYSRSn |
| 13 | IPC13 | R | 0h | Indicates to the local CPU if the IPC13 event flag was set by the remote CPU.<br>0: No IPC13 event was set by the remote CPU<br>1: An IPC13 event was set by the remote CPU<br>Reset type: SYSRSn |
| 12 | IPC12 | R | 0h | Indicates to the local CPU if the IPC12 event flag was set by the remote CPU.<br>0: No IPC12 event was set by the remote CPU<br>1: An IPC12 event was set by the remote CPU<br>Reset type: SYSRSn |
| 11 | IPC11 | R | 0h | Indicates to the local CPU if the IPC11 event flag was set by the remote CPU.<br>0: No IPC11 event was set by the remote CPU<br>1: An IPC11 event was set by the remote CPU<br>Reset type: SYSRSn |
| 10 | IPC10 | R | 0h | Indicates to the local CPU if the IPC10 event flag was set by the remote CPU.<br>0: No IPC10 event was set by the remote CPU<br>1: An IPC10 event was set by the remote CPU<br>Reset type: SYSRSn |
| 9 | IPC9 | R | 0h | Indicates to the local CPU if the IPC9 event flag was set by the remote CPU.<br>0: No IPC9 event was set by the remote CPU<br>1: An IPC9 event was set by the remote CPU<br>Reset type: SYSRSn |
| 8 | IPC8 | R | 0h | Indicates to the local CPU if the IPC8 event flag was set by the remote CPU.<br>0: No IPC8 event was set by the remote CPU<br>1: An IPC8 event was set by the remote CPU<br>Reset type: SYSRSn |
| 7 | IPC7 | R | 0h | Indicates to the local CPU if the IPC7 event flag was set by the remote CPU.<br>0: No IPC7 event was set by the remote CPU<br>1: An IPC7 event was set by the remote CPU<br>Reset type: SYSRSn |
| 6 | IPC6 | R | 0h | Indicates to the local CPU if the IPC6 event flag was set by the remote CPU.<br>0: No IPC6 event was set by the remote CPU<br>1: An IPC6 event was set by the remote CPU<br>Reset type: SYSRSn |
| 5 | IPC5 | R | 0h | Indicates to the local CPU if the IPC5 event flag was set by the remote CPU.<br>0: No IPC5 event was set by the remote CPU<br>1: An IPC5 event was set by the remote CPU<br>Reset type: SYSRSn |
| 4 | IPC4 | R | 0h | Indicates to the local CPU if the IPC4 event flag was set by the remote CPU.<br>0: No IPC4 event was set by the remote CPU<br>1: An IPC4 event was set by the remote CPU<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 7-7. IPCSTS Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3 | IPC3 | R | 0h | Indicates to the local CPU if the IPC3 event flag was set by the remote CPU.<br>0: No IPC3 event was set by the remote CPU<br>1: An IPC3 event was set by the remote CPU<br>Notes<br>[1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.<br>Reset type: SYSRSn |
| 2 | IPC2 | R | 0h | Indicates to the local CPU if the IPC2 event flag was set by the remote CPU.<br>0: No IPC2 event was set by the remote CPU<br>1: An IPC2 event was set by the remote CPU<br>Notes<br>[1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.<br>Reset type: SYSRSn |
| 1 | IPC1 | R | 0h | Indicates to the local CPU if the IPC1 event flag was set by the remote CPU.<br>0: No IPC1 event was set by the remote CPU<br>1: An IPC1 event was set by the remote CPU<br>Notes<br>[1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.<br>Reset type: SYSRSn |
| 0 | IPC0 | R | 0h | Indicates to the local CPU if the IPC0 event flag was set by the remote CPU.<br>0: No IPC0 event was set by the remote CPU<br>1: An IPC0 event was set by the remote CPU<br>Notes<br>[1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 7.7.2.3 IPCSET Register (Offset = 4h) [Reset = 00000000h]

IPCSET is shown in Figure 7-4 and described in Table 7-8.

Return to the Summary Table.

IPC remote flag set register

**Figure 7-4. IPCSET Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| IPC31 | IPC30 | IPC29 | IPC28 | IPC27 | IPC26 | IPC25 | IPC24 |
| R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| IPC23 | IPC22 | IPC21 | IPC20 | IPC19 | IPC18 | IPC17 | IPC16 |
| R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| IPC15 | IPC14 | IPC13 | IPC12 | IPC11 | IPC10 | IPC9 | IPC8 |
| 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 |
| IPC7 | IPC6 | IPC5 | IPC4 | IPC3 | IPC2 | IPC1 | IPC0 |
| 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 7-8. IPCSET Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | IPC31 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC31 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 30 | IPC30 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC30 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 29 | IPC29 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC29 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 28 | IPC28 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC28 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 27 | IPC27 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC27 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 26 | IPC26 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC26 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 25 | IPC25 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC25 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 24 | IPC24 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC24 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 23 | IPC23 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC23 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 22 | IPC22 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC22 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 7-8. IPCSET Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 21 | IPC21 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC21 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 20 | IPC20 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC20 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 19 | IPC19 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC19 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 18 | IPC18 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC18 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 17 | IPC17 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC17 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 16 | IPC16 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC16 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 15 | IPC15 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC15 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 14 | IPC14 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC14 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 13 | IPC13 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC13 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 12 | IPC12 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC12 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 11 | IPC11 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC11 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 10 | IPC10 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC10 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 9 | IPC9 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC9 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 8 | IPC8 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC8 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 7 | IPC7 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC7 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 6 | IPC6 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC6 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 5 | IPC5 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC5 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 7-8. IPCSET Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 4 | IPC4 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC4 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 3 | IPC3 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC3 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.<br>Reset type: SYSRSn |
| 2 | IPC2 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC2 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.<br>Reset type: SYSRSn |
| 1 | IPC1 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC1 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.<br>Reset type: SYSRSn |
| 0 | IPC0 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC0 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 7.7.2.4 IPCCLR Register (Offset = 6h) [Reset = 00000000h]

IPCCLR is shown in Figure 7-5 and described in Table 7-9.

Return to the Summary Table.

IPC remote flag clear register

**Figure 7-5. IPCCLR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| IPC31 | IPC30 | IPC29 | IPC28 | IPC27 | IPC26 | IPC25 | IPC24 |
| R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| IPC23 | IPC22 | IPC21 | IPC20 | IPC19 | IPC18 | IPC17 | IPC16 |
| R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| IPC15 | IPC14 | IPC13 | IPC12 | IPC11 | IPC10 | IPC9 | IPC8 |
| 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 |
| IPC7 | IPC6 | IPC5 | IPC4 | IPC3 | IPC2 | IPC1 | IPC0 |
| 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 7-9. IPCCLR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | IPC31 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC31 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 30 | IPC30 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC30 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 29 | IPC29 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC29 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 28 | IPC28 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC28 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 7-9. IPCCLR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 27 | IPC27 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC27 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 26 | IPC26 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC26 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 25 | IPC25 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC25 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 24 | IPC24 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC24 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 23 | IPC23 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC23 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 22 | IPC22 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC22 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 21 | IPC21 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC21 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 20 | IPC20 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC20 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 7-9. IPCCLR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19 | IPC19 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC19 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 18 | IPC18 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC18 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 17 | IPC17 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC17 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 16 | IPC16 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC16 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 15 | IPC15 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC15 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 14 | IPC14 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC14 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 13 | IPC13 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC13 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 12 | IPC12 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC12 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 7-9. IPCCLR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 11 | IPC11 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC11 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 10 | IPC10 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC10 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 9 | IPC9 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC9 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 8 | IPC8 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC8 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 7 | IPC7 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC7 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 6 | IPC6 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC6 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 5 | IPC5 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC5 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 4 | IPC4 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC4 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 7-9. IPCCLR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3 | IPC3 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC3 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 2 | IPC2 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC2 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 1 | IPC1 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC1 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 0 | IPC0 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC0 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 7.7.2.5 IPCFLG Register (Offset = 8h) [Reset = 00000000h]

IPCFLG is shown in Figure 7-6 and described in Table 7-10.

Return to the Summary Table.

IPC remote flag status register

**Figure 7-6. IPCFLG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| IPC31 | IPC30 | IPC29 | IPC28 | IPC27 | IPC26 | IPC25 | IPC24 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| IPC23 | IPC22 | IPC21 | IPC20 | IPC19 | IPC18 | IPC17 | IPC16 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| IPC15 | IPC14 | IPC13 | IPC12 | IPC11 | IPC10 | IPC9 | IPC8 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| IPC7 | IPC6 | IPC5 | IPC4 | IPC3 | IPC2 | IPC1 | IPC0 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 7-10. IPCFLG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | IPC31 | R | 0h | Indicates to the local CPU whether the remote IPC31 event flag is set.<br>0: The remote IPC31 event flag is not set<br>1: The remote IPC31 event flag is set<br>Reset type: SYSRSn |
| 30 | IPC30 | R | 0h | Indicates to the local CPU whether the remote IPC30 event flag is set.<br>0: The remote IPC30 event flag is not set<br>1: The remote IPC30 event flag is set<br>Reset type: SYSRSn |
| 29 | IPC29 | R | 0h | Indicates to the local CPU whether the remote IPC29 event flag is set.<br>0: The remote IPC29 event flag is not set<br>1: The remote IPC29 event flag is set<br>Reset type: SYSRSn |
| 28 | IPC28 | R | 0h | Indicates to the local CPU whether the remote IPC28 event flag is set.<br>0: The remote IPC28 event flag is not set<br>1: The remote IPC28 event flag is set<br>Reset type: SYSRSn |
| 27 | IPC27 | R | 0h | Indicates to the local CPU whether the remote IPC27 event flag is set.<br>0: The remote IPC27 event flag is not set<br>1: The remote IPC27 event flag is set<br>Reset type: SYSRSn |
| 26 | IPC26 | R | 0h | Indicates to the local CPU whether the remote IPC26 event flag is set.<br>0: The remote IPC26 event flag is not set<br>1: The remote IPC26 event flag is set<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 7-10. IPCFLG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 25 | IPC25 | R | 0h | Indicates to the local CPU whether the remote IPC25 event flag is set.<br>0: The remote IPC25 event flag is not set<br>1: The remote IPC25 event flag is set<br>Reset type: SYSRSn |
| 24 | IPC24 | R | 0h | Indicates to the local CPU whether the remote IPC24 event flag is set.<br>0: The remote IPC24 event flag is not set<br>1: The remote IPC24 event flag is set<br>Reset type: SYSRSn |
| 23 | IPC23 | R | 0h | Indicates to the local CPU whether the remote IPC23 event flag is set.<br>0: The remote IPC23 event flag is not set<br>1: The remote IPC23 event flag is set<br>Reset type: SYSRSn |
| 22 | IPC22 | R | 0h | Indicates to the local CPU whether the remote IPC22 event flag is set.<br>0: The remote IPC22 event flag is not set<br>1: The remote IPC22 event flag is set<br>Reset type: SYSRSn |
| 21 | IPC21 | R | 0h | Indicates to the local CPU whether the remote IPC21 event flag is set.<br>0: The remote IPC21 event flag is not set<br>1: The remote IPC21 event flag is set<br>Reset type: SYSRSn |
| 20 | IPC20 | R | 0h | Indicates to the local CPU whether the remote IPC20 event flag is set.<br>0: The remote IPC20 event flag is not set<br>1: The remote IPC20 event flag is set<br>Reset type: SYSRSn |
| 19 | IPC19 | R | 0h | Indicates to the local CPU whether the remote IPC19 event flag is set.<br>0: The remote IPC19 event flag is not set<br>1: The remote IPC19 event flag is set<br>Reset type: SYSRSn |
| 18 | IPC18 | R | 0h | Indicates to the local CPU whether the remote IPC18 event flag is set.<br>0: The remote IPC18 event flag is not set<br>1: The remote IPC18 event flag is set<br>Reset type: SYSRSn |
| 17 | IPC17 | R | 0h | Indicates to the local CPU whether the remote IPC17 event flag is set.<br>0: The remote IPC17 event flag is not set<br>1: The remote IPC17 event flag is set<br>Reset type: SYSRSn |
| 16 | IPC16 | R | 0h | Indicates to the local CPU whether the remote IPC16 event flag is set.<br>0: The remote IPC16 event flag is not set<br>1: The remote IPC16 event flag is set<br>Reset type: SYSRSn |
| 15 | IPC15 | R | 0h | Indicates to the local CPU whether the remote IPC15 event flag is set.<br>0: The remote IPC15 event flag is not set<br>1: The remote IPC15 event flag is set<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 7-10. IPCFLG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 14 | IPC14 | R | 0h | Indicates to the local CPU whether the remote IPC14 event flag is set.<br>0: The remote IPC14 event flag is not set<br>1: The remote IPC14 event flag is set<br>Reset type: SYSRSn |
| 13 | IPC13 | R | 0h | Indicates to the local CPU whether the remote IPC13 event flag is set.<br>0: The remote IPC13 event flag is not set<br>1: The remote IPC13 event flag is set<br>Reset type: SYSRSn |
| 12 | IPC12 | R | 0h | Indicates to the local CPU whether the remote IPC12 event flag is set.<br>0: The remote IPC12 event flag is not set<br>1: The remote IPC12 event flag is set<br>Reset type: SYSRSn |
| 11 | IPC11 | R | 0h | Indicates to the local CPU whether the remote IPC11 event flag is set.<br>0: The remote IPC11 event flag is not set<br>1: The remote IPC11 event flag is set<br>Reset type: SYSRSn |
| 10 | IPC10 | R | 0h | Indicates to the local CPU whether the remote IPC10 event flag is set.<br>0: The remote IPC10 event flag is not set<br>1: The remote IPC10 event flag is set<br>Reset type: SYSRSn |
| 9 | IPC9 | R | 0h | Indicates to the local CPU whether the remote IPC9 event flag is set.<br>0: The remote IPC9 event flag is not set<br>1: The remote IPC9 event flag is set<br>Reset type: SYSRSn |
| 8 | IPC8 | R | 0h | Indicates to the local CPU whether the remote IPC8 event flag is set.<br>0: The remote IPC8 event flag is not set<br>1: The remote IPC8 event flag is set<br>Reset type: SYSRSn |
| 7 | IPC7 | R | 0h | Indicates to the local CPU whether the remote IPC7 event flag is set.<br>0: The remote IPC7 event flag is not set<br>1: The remote IPC7 event flag is set<br>Reset type: SYSRSn |
| 6 | IPC6 | R | 0h | Indicates to the local CPU whether the remote IPC6 event flag is set.<br>0: The remote IPC6 event flag is not set<br>1: The remote IPC6 event flag is set<br>Reset type: SYSRSn |
| 5 | IPC5 | R | 0h | Indicates to the local CPU whether the remote IPC5 event flag is set.<br>0: The remote IPC5 event flag is not set<br>1: The remote IPC5 event flag is set<br>Reset type: SYSRSn |
| 4 | IPC4 | R | 0h | Indicates to the local CPU whether the remote IPC4 event flag is set.<br>0: The remote IPC4 event flag is not set<br>1: The remote IPC4 event flag is set<br>Reset type: SYSRSn |
| 3 | IPC3 | R | 0h | Indicates to the local CPU whether the remote IPC3 event flag is set.<br>0: The remote IPC3 event flag is not set<br>1: The remote IPC3 event flag is set<br>Notes:<br>[1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 7-10. IPCFLG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 2 | IPC2 | R | 0h | Indicates to the local CPU whether the remote IPC2 event flag is set.<br>0: The remote IPC2 event flag is not set<br>1: The remote IPC2 event flag is set<br>Notes:<br>[1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.<br>Reset type: SYSRSn |
| 1 | IPC1 | R | 0h | Indicates to the local CPU whether the remote IPC1 event flag is set.<br>0: The remote IPC1 event flag is not set<br>1: The remote IPC1 event flag is set<br>Notes:<br>[1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.<br>Reset type: SYSRSn |
| 0 | IPC0 | R | 0h | Indicates to the local CPU whether the remote IPC0 event flag is set.<br>0: The remote IPC0 event flag is not set<br>1: The remote IPC0 event flag is set<br>Notes:<br>[1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 7.7.2.6 IPCCOUNTERL Register (Offset = Ch) [Reset = 00000000h]

IPCCOUNTERL is shown in Figure 7-7 and described in Table 7-11.

Return to the Summary Table.

IPC Counter Low Register

**Figure 7-7. IPCCOUNTERL Register**

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

**Table 7-11. IPCCOUNTERL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | COUNT | R | 0h | This is the lower 32-bits of free running 64 bit timestamp counter clocked by the PLLSYSCLK.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 7.7.2.7 IPCCOUNTERH Register (Offset = Eh) [Reset = 00000000h]

IPCCOUNTERH is shown in Figure 7-8 and described in Table 7-12.

Return to the Summary Table.

IPC Counter High Register

**Figure 7-8. IPCCOUNTERH Register**

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

**Table 7-12. IPCCOUNTERH Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | COUNT | R | 0h | This is the upper 32-bits of free running 64 bit timestamp counter clocked by the PLLSYSCLK.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 7.7.2.8 IPCSENDCOM Register (Offset = 10h) [Reset = 00000000h]

IPCSENDCOM is shown in Figure 7-9 and described in Table 7-13.

Return to the Summary Table.

Local to Remote IPC Command Register

**Figure 7-9. IPCSENDCOM Register**

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

**Table 7-13. IPCSENDCOM Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | COMMAND | R/W | 0h | This is a general purpose register used to send software-defined commands to the remote CPU. It can only be written by the local CPU.<br>Notes<br>[1] The local CPU's IPCSENDCOM is the same physical register as the remote CPU's IPCRECVCOM, and is located at the same address in both CPUs.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 7.7.2.9 IPCSENDADDR Register (Offset = 12h) [Reset = 00000000h]

IPCSENDADDR is shown in Figure 7-10 and described in Table 7-14.

Return to the Summary Table.

Local to Remote IPC Address Register

**Figure 7-10. IPCSENDADDR Register**

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

**Table 7-14. IPCSENDADDR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | ADDRESS | R/W | 0h | This is a general purpose register used to send software-defined addresses to the remote CPU. It can only be written by the local CPU.<br>Notes<br>[1] The local CPU's IPCSENDADDR is the same physical register as the remote CPU's IPCRECVDATA, and is located at the same address in both CPUs.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 7.7.2.10 IPCSENDDATA Register (Offset = 14h) [Reset = 00000000h]

IPCSENDDATA is shown in Figure 7-11 and described in Table 7-15.

Return to the Summary Table.

Local to Remote IPC Data Register

**Figure 7-11. IPCSENDDATA Register**

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

**Table 7-15. IPCSENDDATA Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | WDATA | R/W | 0h | This is a general purpose register used to send software-defined data to the remote CPU. It can only be written by the local CPU.<br>Notes<br>[1] The local CPU's IPCSENDDATA is the same physical register as the remote CPU's IPCRECVDATA, and is located at the same address in both CPUs.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 7.7.2.11 IPCREMOTEREPLY Register (Offset = 16h) [Reset = 00000000h]

IPCREMOTEREPLY is shown in Figure 7-12 and described in Table 7-16.

Return to the Summary Table.

Remote to Local IPC Reply Data Register

**Figure 7-12. IPCREMOTEREPLY Register**

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

**Table 7-16. IPCREMOTEREPLY Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | RDATA | R | 0h | This is a general purpose register used to receive software-defined data from the remote CPU's response to a command. It can only be written by the remote CPU.<br>Notes<br>[1] The local CPU's IPCREMOTEREPLY is the same physical register as the remote CPU's IPCLOCALREPLY, and is located at the same address in both CPUs.<br>[2] This register is reset by a SYRSn of the remote CPU<br>Reset type: CPUx.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 7.7.2.12 IPCRECVCOM Register (Offset = 18h) [Reset = 00000000h]

IPCRECVCOM is shown in Figure 7-13 and described in Table 7-17.

Return to the Summary Table.

Remote to Local IPC Command Register

**Figure 7-13. IPCRECVCOM Register**

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

**Table 7-17. IPCRECVCOM Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | COMMAND | R | 0h | This is a general purpose register used to receive software-defined commands from the remote CPU. It can only be written by the remote CPU.<br>Notes<br>[1] The local CPU's IPCRECVCOM is the same physical register as the remote CPU's IPCSENDCOM, and is located at the same address in both CPUs.<br>[2] This register is reset by a SYRSn of the remote CPU<br>Reset type: CPUx.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 7.7.2.13 IPCRECVADDR Register (Offset = 1Ah) [Reset = 00000000h]

IPCRECVADDR is shown in Figure 7-14 and described in Table 7-18.

Return to the Summary Table.

Remote to Local IPC Address Register

**Figure 7-14. IPCRECVADDR Register**

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

**Table 7-18. IPCRECVADDR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | ADDRESS | R | 0h | This is a general purpose register used to receive software-defined addresses from the remote CPU. It can only be written by the remote CPU.<br>Notes<br>[1] The local CPU's IPCRECVADDR is the same physical register as the remote CPU's IPCSENDADDR, and is located at the same address in both CPUs.<br>[2] This register is reset by a SYRSn of the remote CPU<br>Reset type: CPUx.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 7.7.2.14 IPCRECVDATA Register (Offset = 1Ch) [Reset = 00000000h]

IPCRECVDATA is shown in Figure 7-15 and described in Table 7-19.

Return to the Summary Table.

Remote to Local IPC Data Register

**Figure 7-15. IPCRECVDATA Register**

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

**Table 7-19. IPCRECVDATA Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | WDATA | R | 0h | This is a general purpose register used to receive software-defined data from the remote CPU. It can only be written by the remote CPU.<br>Notes<br>[1] The local CPU's IPCRECVDATA is the same physical register as the remote CPU's IPCSENDDATA, and is located at the same address in both CPUs.<br>[2] This register is reset by a SYRSn of the remote CPU<br>Reset type: CPUx.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 7.7.2.15 IPCLOCALREPLY Register (Offset = 1Eh) [Reset = 00000000h]

IPCLOCALREPLY is shown in Figure 7-16 and described in Table 7-20.

Return to the Summary Table.

Local to Remote IPC Reply Data Register

**Figure 7-16. IPCLOCALREPLY Register**

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

**Table 7-20. IPCLOCALREPLY Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | RDATA | R/W | 0h | This is a general purpose register used to send software-defined data to the remote CPU in response to a command. It can only be written by the local CPU.<br>Notes<br>[1] The local CPU's IPCLOCALREPLY is the same physical register as the remote CPU's IPCREMOTEREPLY, and is located at the same address in both CPUs.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 7.7.2.16 IPCBOOTSTS Register (Offset = 20h) [Reset = 00000000h]

IPCBOOTSTS is shown in Figure 7-17 and described in Table 7-21.

Return to the Summary Table.

CPU2 to CPU1 IPC Boot Status Register

**Figure 7-17. IPCBOOTSTS Register**

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

**Table 7-21. IPCBOOTSTS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | BOOTSTS | R/W | 0h | This register is used by CPU2 to pass the boot Status to CPU1. The data format is software-defined. It can only be written by CPU2.<br>Reset type: CPU2.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 7.7.2.17 IPCBOOTMODE Register (Offset = 22h) [Reset = 00000000h]

IPCBOOTMODE is shown in Figure 7-18 and described in Table 7-22.

Return to the Summary Table.

CPU1 to CPU2 IPC Boot Mode Register

**Figure 7-18. IPCBOOTMODE Register**

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

**Table 7-22. IPCBOOTMODE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | BOOTMODE | R/W | 0h | This register is used by CPU1 to pass a boot mode information to CPU2. The data format is software-defined. It can only be written by CPU1.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 7.7.3 IPC_REGS_CPU2 Registers

Table 7-23 lists the memory-mapped registers for the IPC_REGS_CPU2 registers. All register offset addresses not listed in Table 7-23 should be considered as reserved locations and the register contents should not be modified.

**Table 7-23. IPC_REGS_CPU2 Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | IPCACK | IPC incoming flag clear (acknowledge) register | | Go |
| 2h | IPCSTS | IPC incoming flag status register | | Go |
| 4h | IPCSET | IPC remote flag set register | | Go |
| 6h | IPCCLR | IPC remote flag clear register | | Go |
| 8h | IPCFLG | IPC remote flag status register | | Go |
| Ch | IPCCOUNTERL | IPC Counter Low Register | | Go |
| Eh | IPCCOUNTERH | IPC Counter High Register | | Go |
| 10h | IPCRECVCOM | Remote to Local IPC Command Register | | Go |
| 12h | IPCRECVADDR | Remote to Local IPC Address Register | | Go |
| 14h | IPCRECVDATA | Remote to Local IPC Data Register | | Go |
| 16h | IPCLOCALREPLY | Local to Remote IPC Reply Data Register | | Go |
| 18h | IPCSENDCOM | Local to Remote IPC Command Register | | Go |
| 1Ah | IPCSENDADDR | Local to Remote IPC Address Register | | Go |
| 1Ch | IPCSENDDATA | Local to Remote IPC Data Register | | Go |
| 1Eh | IPCREMOTEREPLY | Remote to Local IPC Reply Data Register | | Go |
| 20h | IPCBOOTSTS | CPU2 to CPU1 IPC Boot Status Register | | Go |
| 22h | IPCBOOTMODE | CPU1 to CPU2 IPC Boot Mode Register | | Go |

Complex bit access types are encoded to fit into small table cells. Table 7-24 shows the codes that are used for access types in this section.

**Table 7-24. IPC_REGS_CPU2 Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| R-0 | R<br>-0 | Read<br>Returns 0s |
| Write Type | | |
| W | W | Write |
| W1S | W<br>1S | Write<br>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. |



Copyright © 2024 Texas Instruments Incorporated

### 7.7.3.1 IPCACK Register (Offset = 0h) [Reset = 00000000h]

IPCACK is shown in Figure 7-19 and described in Table 7-25.

Return to the Summary Table.

IPC incoming flag clear (acknowledge) register

**Figure 7-19. IPCACK Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| IPC31 | IPC30 | IPC29 | IPC28 | IPC27 | IPC26 | IPC25 | IPC24 |
| R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| IPC23 | IPC22 | IPC21 | IPC20 | IPC19 | IPC18 | IPC17 | IPC16 |
| R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| IPC15 | IPC14 | IPC13 | IPC12 | IPC11 | IPC10 | IPC9 | IPC8 |
| 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 |
| IPC7 | IPC6 | IPC5 | IPC4 | IPC3 | IPC2 | IPC1 | IPC0 |
| 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 7-25. IPCACK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | IPC31 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC31 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 30 | IPC30 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC30 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 29 | IPC29 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC29 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 28 | IPC28 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC28 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 27 | IPC27 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC27 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 26 | IPC26 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC26 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 25 | IPC25 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC25 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 7-25. IPCACK Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 24 | IPC24 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC24 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 23 | IPC23 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC23 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 22 | IPC22 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC22 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 21 | IPC21 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC21 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 20 | IPC20 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC20 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 19 | IPC19 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC19 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 18 | IPC18 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC18 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 17 | IPC17 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC17 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 16 | IPC16 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC16 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 15 | IPC15 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC15 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 14 | IPC14 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC14 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 13 | IPC13 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC13 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 12 | IPC12 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC12 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 11 | IPC11 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC11 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 7-25. IPCACK Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 10 | IPC10 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC10 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 9 | IPC9 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC9 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 8 | IPC8 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC8 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 7 | IPC7 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC7 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 6 | IPC6 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC6 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 5 | IPC5 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC5 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 4 | IPC4 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC4 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 3 | IPC3 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC3 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 2 | IPC2 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC2 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 1 | IPC1 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC1 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |
| 0 | IPC0 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC0 event flag which was set by the remote CPU.<br>Writing 0 to this bit has no effect.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 7.7.3.2 IPCSTS Register (Offset = 2h) [Reset = 00000000h]

IPCSTS is shown in Figure 7-20 and described in Table 7-26.

Return to the Summary Table.

IPC incoming flag status register

**Figure 7-20. IPCSTS Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| IPC31 | IPC30 | IPC29 | IPC28 | IPC27 | IPC26 | IPC25 | IPC24 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| IPC23 | IPC22 | IPC21 | IPC20 | IPC19 | IPC18 | IPC17 | IPC16 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| IPC15 | IPC14 | IPC13 | IPC12 | IPC11 | IPC10 | IPC9 | IPC8 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| IPC7 | IPC6 | IPC5 | IPC4 | IPC3 | IPC2 | IPC1 | IPC0 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 7-26. IPCSTS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | IPC31 | R | 0h | Indicates to the local CPU if the IPC31 event flag was set by the remote CPU.<br>0: No IPC31 event was set by the remote CPU<br>1: An IPC31 event was set by the remote CPU<br>Reset type: SYSRSn |
| 30 | IPC30 | R | 0h | Indicates to the local CPU if the IPC30 event flag was set by the remote CPU.<br>0: No IPC30 event was set by the remote CPU<br>1: An IPC30 event was set by the remote CPU<br>Reset type: SYSRSn |
| 29 | IPC29 | R | 0h | Indicates to the local CPU if the IPC29 event flag was set by the remote CPU.<br>0: No IPC29 event was set by the remote CPU<br>1: An IPC29 event was set by the remote CPU<br>Reset type: SYSRSn |
| 28 | IPC28 | R | 0h | Indicates to the local CPU if the IPC28 event flag was set by the remote CPU.<br>0: No IPC28 event was set by the remote CPU<br>1: An IPC28 event was set by the remote CPU<br>Reset type: SYSRSn |
| 27 | IPC27 | R | 0h | Indicates to the local CPU if the IPC27 event flag was set by the remote CPU.<br>0: No IPC27 event was set by the remote CPU<br>1: An IPC27 event was set by the remote CPU<br>Reset type: SYSRSn |
| 26 | IPC26 | R | 0h | Indicates to the local CPU if the IPC26 event flag was set by the remote CPU.<br>0: No IPC26 event was set by the remote CPU<br>1: An IPC26 event was set by the remote CPU<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 7-26. IPCSTS Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 25 | IPC25 | R | 0h | Indicates to the local CPU if the IPC25 event flag was set by the remote CPU.<br>0: No IPC25 event was set by the remote CPU<br>1: An IPC25 event was set by the remote CPU<br>Reset type: SYSRSn |
| 24 | IPC24 | R | 0h | Indicates to the local CPU if the IPC24 event flag was set by the remote CPU.<br>0: No IPC24 event was set by the remote CPU<br>1: An IPC24 event was set by the remote CPU<br>Reset type: SYSRSn |
| 23 | IPC23 | R | 0h | Indicates to the local CPU if the IPC23 event flag was set by the remote CPU.<br>0: No IPC23 event was set by the remote CPU<br>1: An IPC23 event was set by the remote CPU<br>Reset type: SYSRSn |
| 22 | IPC22 | R | 0h | Indicates to the local CPU if the IPC22 event flag was set by the remote CPU.<br>0: No IPC22 event was set by the remote CPU<br>1: An IPC22 event was set by the remote CPU<br>Reset type: SYSRSn |
| 21 | IPC21 | R | 0h | Indicates to the local CPU if the IPC21 event flag was set by the remote CPU.<br>0: No IPC21 event was set by the remote CPU<br>1: An IPC21 event was set by the remote CPU<br>Reset type: SYSRSn |
| 20 | IPC20 | R | 0h | Indicates to the local CPU if the IPC20 event flag was set by the remote CPU.<br>0: No IPC20 event was set by the remote CPU<br>1: An IPC20 event was set by the remote CPU<br>Reset type: SYSRSn |
| 19 | IPC19 | R | 0h | Indicates to the local CPU if the IPC19 event flag was set by the remote CPU.<br>0: No IPC19 event was set by the remote CPU<br>1: An IPC19 event was set by the remote CPU<br>Reset type: SYSRSn |
| 18 | IPC18 | R | 0h | Indicates to the local CPU if the IPC18 event flag was set by the remote CPU.<br>0: No IPC18 event was set by the remote CPU<br>1: An IPC18 event was set by the remote CPU<br>Reset type: SYSRSn |
| 17 | IPC17 | R | 0h | Indicates to the local CPU if the IPC17 event flag was set by the remote CPU.<br>0: No IPC17 event was set by the remote CPU<br>1: An IPC17 event was set by the remote CPU<br>Reset type: SYSRSn |
| 16 | IPC16 | R | 0h | Indicates to the local CPU if the IPC16 event flag was set by the remote CPU.<br>0: No IPC16 event was set by the remote CPU<br>1: An IPC16 event was set by the remote CPU<br>Reset type: SYSRSn |
| 15 | IPC15 | R | 0h | Indicates to the local CPU if the IPC15 event flag was set by the remote CPU.<br>0: No IPC15 event was set by the remote CPU<br>1: An IPC15 event was set by the remote CPU<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 7-26. IPCSTS Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 14 | IPC14 | R | 0h | Indicates to the local CPU if the IPC14 event flag was set by the remote CPU.<br>0: No IPC14 event was set by the remote CPU<br>1: An IPC14 event was set by the remote CPU<br>Reset type: SYSRSn |
| 13 | IPC13 | R | 0h | Indicates to the local CPU if the IPC13 event flag was set by the remote CPU.<br>0: No IPC13 event was set by the remote CPU<br>1: An IPC13 event was set by the remote CPU<br>Reset type: SYSRSn |
| 12 | IPC12 | R | 0h | Indicates to the local CPU if the IPC12 event flag was set by the remote CPU.<br>0: No IPC12 event was set by the remote CPU<br>1: An IPC12 event was set by the remote CPU<br>Reset type: SYSRSn |
| 11 | IPC11 | R | 0h | Indicates to the local CPU if the IPC11 event flag was set by the remote CPU.<br>0: No IPC11 event was set by the remote CPU<br>1: An IPC11 event was set by the remote CPU<br>Reset type: SYSRSn |
| 10 | IPC10 | R | 0h | Indicates to the local CPU if the IPC10 event flag was set by the remote CPU.<br>0: No IPC10 event was set by the remote CPU<br>1: An IPC10 event was set by the remote CPU<br>Reset type: SYSRSn |
| 9 | IPC9 | R | 0h | Indicates to the local CPU if the IPC9 event flag was set by the remote CPU.<br>0: No IPC9 event was set by the remote CPU<br>1: An IPC9 event was set by the remote CPU<br>Reset type: SYSRSn |
| 8 | IPC8 | R | 0h | Indicates to the local CPU if the IPC8 event flag was set by the remote CPU.<br>0: No IPC8 event was set by the remote CPU<br>1: An IPC8 event was set by the remote CPU<br>Reset type: SYSRSn |
| 7 | IPC7 | R | 0h | Indicates to the local CPU if the IPC7 event flag was set by the remote CPU.<br>0: No IPC7 event was set by the remote CPU<br>1: An IPC7 event was set by the remote CPU<br>Reset type: SYSRSn |
| 6 | IPC6 | R | 0h | Indicates to the local CPU if the IPC6 event flag was set by the remote CPU.<br>0: No IPC6 event was set by the remote CPU<br>1: An IPC6 event was set by the remote CPU<br>Reset type: SYSRSn |
| 5 | IPC5 | R | 0h | Indicates to the local CPU if the IPC5 event flag was set by the remote CPU.<br>0: No IPC5 event was set by the remote CPU<br>1: An IPC5 event was set by the remote CPU<br>Reset type: SYSRSn |
| 4 | IPC4 | R | 0h | Indicates to the local CPU if the IPC4 event flag was set by the remote CPU.<br>0: No IPC4 event was set by the remote CPU<br>1: An IPC4 event was set by the remote CPU<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 7-26. IPCSTS Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3 | IPC3 | R | 0h | Indicates to the local CPU if the IPC3 event flag was set by the remote CPU.<br>0: No IPC3 event was set by the remote CPU<br>1: An IPC3 event was set by the remote CPU<br>Notes<br>[1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.<br>Reset type: SYSRSn |
| 2 | IPC2 | R | 0h | Indicates to the local CPU if the IPC2 event flag was set by the remote CPU.<br>0: No IPC2 event was set by the remote CPU<br>1: An IPC2 event was set by the remote CPU<br>Notes<br>[1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.<br>Reset type: SYSRSn |
| 1 | IPC1 | R | 0h | Indicates to the local CPU if the IPC1 event flag was set by the remote CPU.<br>0: No IPC1 event was set by the remote CPU<br>1: An IPC1 event was set by the remote CPU<br>Notes<br>[1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.<br>Reset type: SYSRSn |
| 0 | IPC0 | R | 0h | Indicates to the local CPU if the IPC0 event flag was set by the remote CPU.<br>0: No IPC0 event was set by the remote CPU<br>1: An IPC0 event was set by the remote CPU<br>Notes<br>[1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 7.7.3.3 IPCSET Register (Offset = 4h) [Reset = 00000000h]

IPCSET is shown in Figure 7-21 and described in Table 7-27.

Return to the Summary Table.

IPC remote flag set register

**Figure 7-21. IPCSET Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| IPC31 | IPC30 | IPC29 | IPC28 | IPC27 | IPC26 | IPC25 | IPC24 |
| R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| IPC23 | IPC22 | IPC21 | IPC20 | IPC19 | IPC18 | IPC17 | IPC16 |
| R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| IPC15 | IPC14 | IPC13 | IPC12 | IPC11 | IPC10 | IPC9 | IPC8 |
| 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 |
| IPC7 | IPC6 | IPC5 | IPC4 | IPC3 | IPC2 | IPC1 | IPC0 |
| 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 7-27. IPCSET Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | IPC31 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC31 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 30 | IPC30 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC30 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 29 | IPC29 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC29 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 28 | IPC28 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC28 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 27 | IPC27 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC27 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 26 | IPC26 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC26 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 25 | IPC25 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC25 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 24 | IPC24 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC24 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 23 | IPC23 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC23 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 22 | IPC22 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC22 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 7-27. IPCSET Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 21 | IPC21 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC21 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 20 | IPC20 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC20 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 19 | IPC19 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC19 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 18 | IPC18 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC18 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 17 | IPC17 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC17 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 16 | IPC16 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC16 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 15 | IPC15 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC15 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 14 | IPC14 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC14 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 13 | IPC13 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC13 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 12 | IPC12 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC12 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 11 | IPC11 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC11 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 10 | IPC10 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC10 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 9 | IPC9 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC9 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 8 | IPC8 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC8 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 7 | IPC7 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC7 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 6 | IPC6 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC6 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 5 | IPC5 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC5 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 7-27. IPCSET Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 4 | IPC4 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC4 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 3 | IPC3 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC3 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.<br>Reset type: SYSRSn |
| 2 | IPC2 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC2 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.<br>Reset type: SYSRSn |
| 1 | IPC1 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC1 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.<br>Reset type: SYSRSn |
| 0 | IPC0 | R-0/W1S | 0h | Writing 1 to this bit sets the IPC0 event flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 7.7.3.4 IPCCLR Register (Offset = 6h) [Reset = 00000000h]

IPCCLR is shown in Figure 7-22 and described in Table 7-28.

Return to the Summary Table.

IPC remote flag clear register

**Figure 7-22. IPCCLR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| IPC31 | IPC30 | IPC29 | IPC28 | IPC27 | IPC26 | IPC25 | IPC24 |
| R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| IPC23 | IPC22 | IPC21 | IPC20 | IPC19 | IPC18 | IPC17 | IPC16 |
| R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| IPC15 | IPC14 | IPC13 | IPC12 | IPC11 | IPC10 | IPC9 | IPC8 |
| 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 |
| IPC7 | IPC6 | IPC5 | IPC4 | IPC3 | IPC2 | IPC1 | IPC0 |
| 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 7-28. IPCCLR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | IPC31 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC31 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 30 | IPC30 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC30 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 29 | IPC29 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC29 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 28 | IPC28 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC28 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 7-28. IPCCLR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 27 | IPC27 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC27 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 26 | IPC26 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC26 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 25 | IPC25 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC25 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 24 | IPC24 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC24 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 23 | IPC23 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC23 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 22 | IPC22 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC22 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 21 | IPC21 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC21 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 20 | IPC20 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC20 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 7-28. IPCCLR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19 | IPC19 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC19 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 18 | IPC18 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC18 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 17 | IPC17 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC17 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 16 | IPC16 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC16 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 15 | IPC15 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC15 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 14 | IPC14 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC14 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 13 | IPC13 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC13 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 12 | IPC12 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC12 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 7-28. IPCCLR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 11 | IPC11 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC11 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 10 | IPC10 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC10 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 9 | IPC9 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC9 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 8 | IPC8 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC8 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 7 | IPC7 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC7 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 6 | IPC6 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC6 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 5 | IPC5 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC5 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 4 | IPC4 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC4 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 7-28. IPCCLR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3 | IPC3 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC3 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 2 | IPC2 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC2 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 1 | IPC1 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC1 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |
| 0 | IPC0 | R-0/W1S | 0h | Writing 1 to this bit clears the IPC0 flag for the remote CPU.<br>Writing 0 has no effect.<br>Notes:<br>[1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 7.7.3.5 IPCFLG Register (Offset = 8h) [Reset = 00000000h]

IPCFLG is shown in Figure 7-23 and described in Table 7-29.

Return to the Summary Table.

IPC remote flag status register

**Figure 7-23. IPCFLG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| IPC31 | IPC30 | IPC29 | IPC28 | IPC27 | IPC26 | IPC25 | IPC24 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| IPC23 | IPC22 | IPC21 | IPC20 | IPC19 | IPC18 | IPC17 | IPC16 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| IPC15 | IPC14 | IPC13 | IPC12 | IPC11 | IPC10 | IPC9 | IPC8 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| IPC7 | IPC6 | IPC5 | IPC4 | IPC3 | IPC2 | IPC1 | IPC0 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 7-29. IPCFLG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | IPC31 | R | 0h | Indicates to the local CPU whether the remote IPC31 event flag is set.<br>0: The remote IPC31 event flag is not set<br>1: The remote IPC31 event flag is set<br>Reset type: SYSRSn |
| 30 | IPC30 | R | 0h | Indicates to the local CPU whether the remote IPC30 event flag is set.<br>0: The remote IPC30 event flag is not set<br>1: The remote IPC30 event flag is set<br>Reset type: SYSRSn |
| 29 | IPC29 | R | 0h | Indicates to the local CPU whether the remote IPC29 event flag is set.<br>0: The remote IPC29 event flag is not set<br>1: The remote IPC29 event flag is set<br>Reset type: SYSRSn |
| 28 | IPC28 | R | 0h | Indicates to the local CPU whether the remote IPC28 event flag is set.<br>0: The remote IPC28 event flag is not set<br>1: The remote IPC28 event flag is set<br>Reset type: SYSRSn |
| 27 | IPC27 | R | 0h | Indicates to the local CPU whether the remote IPC27 event flag is set.<br>0: The remote IPC27 event flag is not set<br>1: The remote IPC27 event flag is set<br>Reset type: SYSRSn |
| 26 | IPC26 | R | 0h | Indicates to the local CPU whether the remote IPC26 event flag is set.<br>0: The remote IPC26 event flag is not set<br>1: The remote IPC26 event flag is set<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 7-29. IPCFLG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 25 | IPC25 | R | 0h | Indicates to the local CPU whether the remote IPC25 event flag is set.<br>0: The remote IPC25 event flag is not set<br>1: The remote IPC25 event flag is set<br>Reset type: SYSRSn |
| 24 | IPC24 | R | 0h | Indicates to the local CPU whether the remote IPC24 event flag is set.<br>0: The remote IPC24 event flag is not set<br>1: The remote IPC24 event flag is set<br>Reset type: SYSRSn |
| 23 | IPC23 | R | 0h | Indicates to the local CPU whether the remote IPC23 event flag is set.<br>0: The remote IPC23 event flag is not set<br>1: The remote IPC23 event flag is set<br>Reset type: SYSRSn |
| 22 | IPC22 | R | 0h | Indicates to the local CPU whether the remote IPC22 event flag is set.<br>0: The remote IPC22 event flag is not set<br>1: The remote IPC22 event flag is set<br>Reset type: SYSRSn |
| 21 | IPC21 | R | 0h | Indicates to the local CPU whether the remote IPC21 event flag is set.<br>0: The remote IPC21 event flag is not set<br>1: The remote IPC21 event flag is set<br>Reset type: SYSRSn |
| 20 | IPC20 | R | 0h | Indicates to the local CPU whether the remote IPC20 event flag is set.<br>0: The remote IPC20 event flag is not set<br>1: The remote IPC20 event flag is set<br>Reset type: SYSRSn |
| 19 | IPC19 | R | 0h | Indicates to the local CPU whether the remote IPC19 event flag is set.<br>0: The remote IPC19 event flag is not set<br>1: The remote IPC19 event flag is set<br>Reset type: SYSRSn |
| 18 | IPC18 | R | 0h | Indicates to the local CPU whether the remote IPC18 event flag is set.<br>0: The remote IPC18 event flag is not set<br>1: The remote IPC18 event flag is set<br>Reset type: SYSRSn |
| 17 | IPC17 | R | 0h | Indicates to the local CPU whether the remote IPC17 event flag is set.<br>0: The remote IPC17 event flag is not set<br>1: The remote IPC17 event flag is set<br>Reset type: SYSRSn |
| 16 | IPC16 | R | 0h | Indicates to the local CPU whether the remote IPC16 event flag is set.<br>0: The remote IPC16 event flag is not set<br>1: The remote IPC16 event flag is set<br>Reset type: SYSRSn |
| 15 | IPC15 | R | 0h | Indicates to the local CPU whether the remote IPC15 event flag is set.<br>0: The remote IPC15 event flag is not set<br>1: The remote IPC15 event flag is set<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 7-29. IPCFLG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 14 | IPC14 | R | 0h | Indicates to the local CPU whether the remote IPC14 event flag is set.<br>0: The remote IPC14 event flag is not set<br>1: The remote IPC14 event flag is set<br>Reset type: SYSRSn |
| 13 | IPC13 | R | 0h | Indicates to the local CPU whether the remote IPC13 event flag is set.<br>0: The remote IPC13 event flag is not set<br>1: The remote IPC13 event flag is set<br>Reset type: SYSRSn |
| 12 | IPC12 | R | 0h | Indicates to the local CPU whether the remote IPC12 event flag is set.<br>0: The remote IPC12 event flag is not set<br>1: The remote IPC12 event flag is set<br>Reset type: SYSRSn |
| 11 | IPC11 | R | 0h | Indicates to the local CPU whether the remote IPC11 event flag is set.<br>0: The remote IPC11 event flag is not set<br>1: The remote IPC11 event flag is set<br>Reset type: SYSRSn |
| 10 | IPC10 | R | 0h | Indicates to the local CPU whether the remote IPC10 event flag is set.<br>0: The remote IPC10 event flag is not set<br>1: The remote IPC10 event flag is set<br>Reset type: SYSRSn |
| 9 | IPC9 | R | 0h | Indicates to the local CPU whether the remote IPC9 event flag is set.<br>0: The remote IPC9 event flag is not set<br>1: The remote IPC9 event flag is set<br>Reset type: SYSRSn |
| 8 | IPC8 | R | 0h | Indicates to the local CPU whether the remote IPC8 event flag is set.<br>0: The remote IPC8 event flag is not set<br>1: The remote IPC8 event flag is set<br>Reset type: SYSRSn |
| 7 | IPC7 | R | 0h | Indicates to the local CPU whether the remote IPC7 event flag is set.<br>0: The remote IPC7 event flag is not set<br>1: The remote IPC7 event flag is set<br>Reset type: SYSRSn |
| 6 | IPC6 | R | 0h | Indicates to the local CPU whether the remote IPC6 event flag is set.<br>0: The remote IPC6 event flag is not set<br>1: The remote IPC6 event flag is set<br>Reset type: SYSRSn |
| 5 | IPC5 | R | 0h | Indicates to the local CPU whether the remote IPC5 event flag is set.<br>0: The remote IPC5 event flag is not set<br>1: The remote IPC5 event flag is set<br>Reset type: SYSRSn |
| 4 | IPC4 | R | 0h | Indicates to the local CPU whether the remote IPC4 event flag is set.<br>0: The remote IPC4 event flag is not set<br>1: The remote IPC4 event flag is set<br>Reset type: SYSRSn |
| 3 | IPC3 | R | 0h | Indicates to the local CPU whether the remote IPC3 event flag is set.<br>0: The remote IPC3 event flag is not set<br>1: The remote IPC3 event flag is set<br>Notes:<br>[1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 7-29. IPCFLG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 2 | IPC2 | R | 0h | Indicates to the local CPU whether the remote IPC2 event flag is set.<br>0: The remote IPC2 event flag is not set<br>1: The remote IPC2 event flag is set<br>Notes:<br>[1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.<br>Reset type: SYSRSn |
| 1 | IPC1 | R | 0h | Indicates to the local CPU whether the remote IPC1 event flag is set.<br>0: The remote IPC1 event flag is not set<br>1: The remote IPC1 event flag is set<br>Notes:<br>[1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.<br>Reset type: SYSRSn |
| 0 | IPC0 | R | 0h | Indicates to the local CPU whether the remote IPC0 event flag is set.<br>0: The remote IPC0 event flag is not set<br>1: The remote IPC0 event flag is set<br>Notes:<br>[1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 7.7.3.6 IPCCOUNTERL Register (Offset = Ch) [Reset = 00000000h]

IPCCOUNTERL is shown in Figure 7-24 and described in Table 7-30.

Return to the Summary Table.

IPC Counter Low Register

**Figure 7-24. IPCCOUNTERL Register**

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

**Table 7-30. IPCCOUNTERL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | COUNT | R | 0h | This is the lower 32-bits of free running 64 bit timestamp counter clocked by the PLLSYSCLK.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 7.7.3.7 IPCCOUNTERH Register (Offset = Eh) [Reset = 00000000h]

IPCCOUNTERH is shown in Figure 7-25 and described in Table 7-31.

Return to the Summary Table.

IPC Counter High Register

**Figure 7-25. IPCCOUNTERH Register**

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

**Table 7-31. IPCCOUNTERH Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | COUNT | R | 0h | This is the upper 32-bits of free running 64 bit timestamp counter clocked by the PLLSYSCLK.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 7.7.3.8 IPCRECVCOM Register (Offset = 10h) [Reset = 00000000h]

IPCRECVCOM is shown in Figure 7-26 and described in Table 7-32.

Return to the Summary Table.

Remote to Local IPC Command Register

**Figure 7-26. IPCRECVCOM Register**

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

**Table 7-32. IPCRECVCOM Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | COMMAND | R | 0h | This is a general purpose register used to receive software-defined commands from the remote CPU. It can only be written by the remote CPU.<br>Notes<br>[1] The local CPU's IPCRECVCOM is the same physical register as the remote CPU's IPCSENDCOM, and is located at the same address in both CPUs.<br>[2] This register is reset by a SYRSn of the remote CPU<br>Reset type: CPUx.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 7.7.3.9 IPCRECVADDR Register (Offset = 12h) [Reset = 00000000h]

IPCRECVADDR is shown in Figure 7-27 and described in Table 7-33.

Return to the Summary Table.

Remote to Local IPC Address Register

**Figure 7-27. IPCRECVADDR Register**

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

**Table 7-33. IPCRECVADDR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | ADDRESS | R | 0h | This is a general purpose register used to receive software-defined addresses from the remote CPU. It can only be written by the remote CPU.<br>Notes<br>[1] The local CPU's IPCRECVADDR is the same physical register as the remote CPU's IPCSENDADDR, and is located at the same address in both CPUs.<br>[2] This register is reset by a SYRSn of the remote CPU<br>Reset type: CPUx.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 7.7.3.10 IPCRECVDATA Register (Offset = 14h) [Reset = 00000000h]

IPCRECVDATA is shown in Figure 7-28 and described in Table 7-34.

Return to the Summary Table.

Remote to Local IPC Data Register

**Figure 7-28. IPCRECVDATA Register**

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

**Table 7-34. IPCRECVDATA Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | WDATA | R | 0h | This is a general purpose register used to receive software-defined data from the remote CPU. It can only be written by the remote CPU.<br>Notes<br>[1] The local CPU's IPCRECVDATA is the same physical register as the remote CPU's IPCSENDDATA, and is located at the same address in both CPUs.<br>[2] This register is reset by a SYRSn of the remote CPU<br>Reset type: CPUx.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 7.7.3.11 IPCLOCALREPLY Register (Offset = 16h) [Reset = 00000000h]

IPCLOCALREPLY is shown in Figure 7-29 and described in Table 7-35.

Return to the Summary Table.

Local to Remote IPC Reply Data Register

**Figure 7-29. IPCLOCALREPLY Register**

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

**Table 7-35. IPCLOCALREPLY Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | RDATA | R/W | 0h | This is a general purpose register used to send software-defined data to the remote CPU in response to a command. It can only be written by the local CPU.<br>Notes<br>[1] The local CPU's IPCLOCALREPLY is the same physical register as the remote CPU's IPCREMOTEREPLY, and is located at the same address in both CPUs.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 7.7.3.12 IPCSENDCOM Register (Offset = 18h) [Reset = 00000000h]

IPCSENDCOM is shown in Figure 7-30 and described in Table 7-36.

Return to the Summary Table.

Local to Remote IPC Command Register

**Figure 7-30. IPCSENDCOM Register**

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

**Table 7-36. IPCSENDCOM Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | COMMAND | R/W | 0h | This is a general purpose register used to send software-defined commands to the remote CPU. It can only be written by the local CPU.<br>Notes<br>[1] The local CPU's IPCSENDCOM is the same physical register as the remote CPU's IPCRECVCOM, and is located at the same address in both CPUs.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 7.7.3.13 IPCSENDADDR Register (Offset = 1Ah) [Reset = 00000000h]

IPCSENDADDR is shown in Figure 7-31 and described in Table 7-37.

Return to the Summary Table.

Local to Remote IPC Address Register

**Figure 7-31. IPCSENDADDR Register**

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

**Table 7-37. IPCSENDADDR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | ADDRESS | R/W | 0h | This is a general purpose register used to send software-defined addresses to the remote CPU. It can only be written by the local CPU.<br>Notes<br>[1] The local CPU's IPCSENDADDR is the same physical register as the remote CPU's IPCRECVDATA, and is located at the same address in both CPUs.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 7.7.3.14 IPCSENDDATA Register (Offset = 1Ch) [Reset = 00000000h]

IPCSENDDATA is shown in Figure 7-32 and described in Table 7-38.

Return to the Summary Table.

Local to Remote IPC Data Register

**Figure 7-32. IPCSENDDATA Register**

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

**Table 7-38. IPCSENDDATA Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | WDATA | R/W | 0h | This is a general purpose register used to send software-defined data to the remote CPU. It can only be written by the local CPU.<br>Notes<br>[1] The local CPU's IPCSENDDATA is the same physical register as the remote CPU's IPCRECVDATA, and is located at the same address in both CPUs.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 7.7.3.15 IPCREMOTEREPLY Register (Offset = 1Eh) [Reset = 00000000h]

IPCREMOTEREPLY is shown in Figure 7-33 and described in Table 7-39.

Return to the Summary Table.

Remote to Local IPC Reply Data Register

**Figure 7-33. IPCREMOTEREPLY Register**

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

**Table 7-39. IPCREMOTEREPLY Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | RDATA | R | 0h | This is a general purpose register used to receive software-defined data from the remote CPU's response to a command. It can only be written by the remote CPU.<br>Notes<br>[1] The local CPU's IPCREMOTEREPLY is the same physical register as the remote CPU's IPCLOCALREPLY, and is located at the same address in both CPUs.<br>[2] This register is reset by a SYRSn of the remote CPU<br>Reset type: CPUx.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 7.7.3.16 IPCBOOTSTS Register (Offset = 20h) [Reset = 00000000h]

IPCBOOTSTS is shown in Figure 7-34 and described in Table 7-40.

Return to the Summary Table.

CPU2 to CPU1 IPC Boot Status Register

**Figure 7-34. IPCBOOTSTS Register**

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

**Table 7-40. IPCBOOTSTS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | BOOTSTS | R/W | 0h | This register is used by CPU2 to pass the boot Status to CPU1. The data format is software-defined. It can only be written by CPU2.<br>Reset type: CPU2.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 7.7.3.17 IPCBOOTMODE Register (Offset = 22h) [Reset = 00000000h]

IPCBOOTMODE is shown in Figure 7-35 and described in Table 7-41.

Return to the Summary Table.

CPU1 to CPU2 IPC Boot Mode Register

**Figure 7-35. IPCBOOTMODE Register**

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

**Table 7-41. IPCBOOTMODE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | BOOTMODE | R/W | 0h | This register is used by CPU1 to pass a boot mode information to CPU2. The data format is software-defined. It can only be written by CPU1.<br>Reset type: CPU1.SYSRSn |

### *7.7.4 IPC Registers to Driverlib Functions*

**Table 7-42. IPC Registers to Driverlib Functions**

| File | Driverlib Function |
|---|---|
| **ACK** | |
| - | |
| **STS** | |
| - | |
| **SET** | |
| - | |
| **CLR** | |
| - | |
| **FLG** | |
| - | |
| **COUNTERL** | |
| - | |
| **COUNTERH** | |
| - | |
| **SENDCOM** | |
| - | |
| **SENDADDR** | |
| - | |
| **SENDDATA** | |
| - | |
| **REMOTEREPLY** | |
| - | |
| **RECVCOM** | |
| - | |
| **RECVADDR** | |
| - | |
| **RECVDATA** | |
| - | |



Copyright © 2024 Texas Instruments Incorporated

**Table 7-42. IPC Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| **LOCALREPLY** | |
| - | |
| **RECVCOM** | |
| - | |
| **RECVADDR** | |
| - | |
| **RECVDATA** | |
| - | |
| **LOCALREPLY** | |
| - | |
| **SENDCOM** | |
| - | |
| **SENDADDR** | |
| - | |
| **SENDDATA** | |
| - | |
| **REMOTEREPLY** | |
| - | |
| **BOOTSTS** | |
| - | |
| **BOOTMODE** | |
| - | |



Copyright © 2024 Texas Instruments Incorporated

Chapter 8

# General-Purpose Input/Output (GPIO)

The GPIO module controls the device's digital 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.


**8.1 Introduction** ..... 987
**8.2 Configuration Overview** ..... 989
**8.3 Digital General-Purpose I/O Control** ..... 990
**8.4 Input Qualification** ..... 991
**8.5 USB Signals** ..... 995
**8.6 SPI Signals** ..... 995
**8.7 GPIO and Peripheral Muxing** ..... 996
**8.8 Internal Pullup Configuration Requirements** ..... 1004
**8.9 Software** ..... 1005
**8.10 GPIO Registers** ..... 1006





Copyright © 2024 Texas Instruments Incorporated

## 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 four CPU masters.

- CPU1
- CPU1.CLA
- CPU2
- CPU2.CLA

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.

Figure 8-1 shows the GPIO logic for a single pin.

**Note**

The USB PHY pin muxing is not shown in Figure 8-1. For more details on USB pins, see Section 8.5.

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 both CPUs and CLAs to read the physical state of the pin independent of CPU mastering and 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 masters and peripherals. However, the peripheral muxing, CPU muxing, and pin options can only be configured by CPU1.

A separate configuration is required for the USB signals. See Section 8.5 for details.



Copyright © 2024 Texas Instruments Incorporated

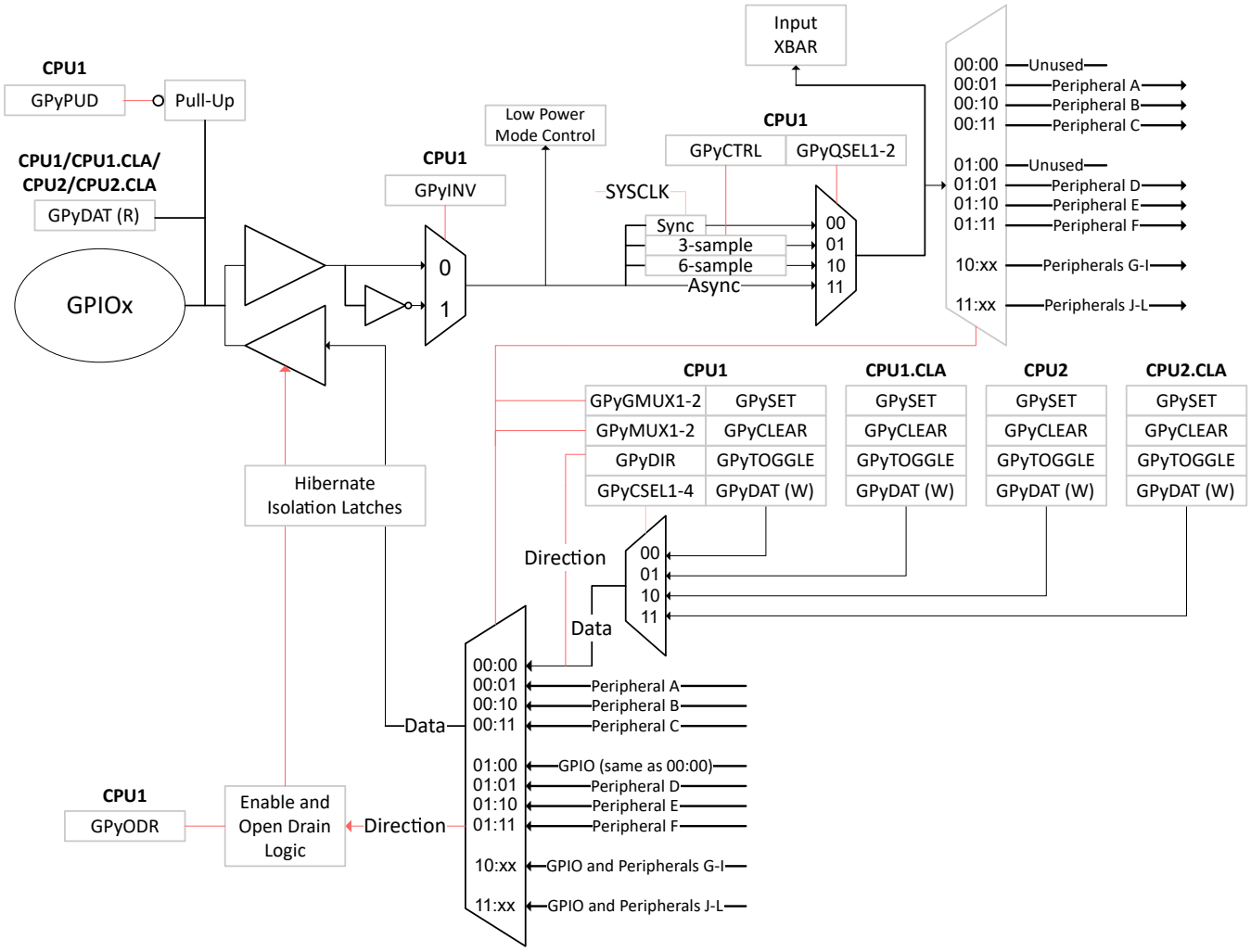



**Figure 8-1. GPIO Logic for a Single Pin**

### *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



Copyright © 2024 Texas Instruments Incorporated

## 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 for each CPU and CLA.

   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.
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. For an explanation of input qualification, see Section 8.4.
4. **Select the direction of any general-purpose I/O pins:** For each pin configured as a GPIO, specify the direction of the pin as either input or output using the GPyDIR registers. By default, all GPIO pins are inputs. Before changing a pin to an output, load the output latch with the value to be driven by writing that value to the GPySET, GPyCLEAR, or GPyDAT registers. Once the latch is loaded, write to GPyDIR to change the pin direction. By default, all output latches are zero.
5. **Select low-power mode wake-up sources:** GPIOs 0-63 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. In Hibernate mode, GPIO 41 is the only wake-up pin. For more information on low-power modes and GPIO wake-up, see the Low-Power Modes section in the *System Control and Interrupts* chapter.
6. **Select external interrupt sources:** Configuring external interrupts is a two-step process. First, the interrupts themselves must be enabled and 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.

**Note**

Configure the GPIO registers GPxMUX1, GPxMUX2, GPxINV, GPxGMUX1, and GPxGMUX2 as per Section 8.7 before a peripheral starts using the respective GPIOs. The configuration is expected to be static during runtime.



Copyright © 2024 Texas Instruments Incorporated

## 8.3 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 GPIOA1 by writing to the GPADAT register bit 0 using a read-modify-write instruction, a problem can occur if another I/O port A signal changes level between the read and the write stage of the instruction. Following is an analysis of why this happens:

  The GPyDAT registers reflect the state of the pin, not the latch. This means the register reflects the actual pin value. However, there is a lag between when the register is written to when the new pin value is reflected back in the register. This 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.

- **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.



Copyright © 2024 Texas Instruments Incorporated

- **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.

## 8.4 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.

### *8.4.1 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 $I^2C$. 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.

### *8.4.2 Synchronization to SYSCLKOUT Only*

This is the default qualification mode of all the pins at reset. In this mode, the input signal is only synchronized to the system clock (SYSCLKOUT). Because the incoming signal is asynchronous, a SYSCLKOUT period of delay is needed for the input to the device to be changed. No further qualification is performed on the signal.



Copyright © 2024 Texas Instruments Incorporated

### 8.4.3 Qualification Using a Sampling Window

In this mode, the signal is first synchronized to the system clock (SYSCLKOUT) and then qualified by a specified number of cycles before the input is allowed to change. Figure 8-2 and Figure 8-3 show how the input qualification is performed to eliminate unwanted noise. Two parameters are specified by the user for this type of qualification: 1) the sampling period, or how often the signal is sampled, and 2) the number of samples to be taken.

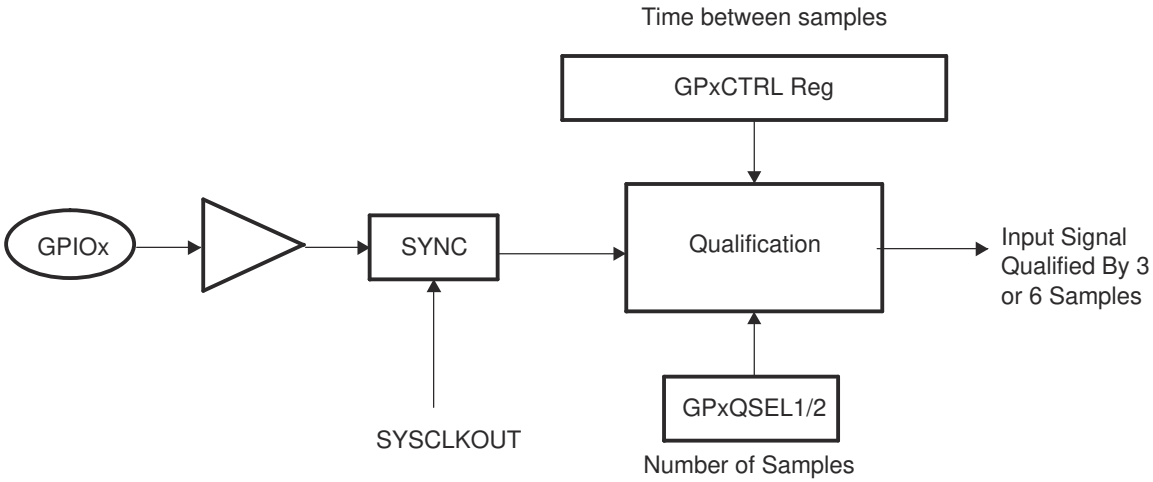



**Figure 8-2. Input Qualification Using a Sampling Window**

**Time between samples (sampling period):**

To qualify the signal, the input signal is sampled at a regular period. The sampling period is specified by the user and determines the time duration between samples, or how often the signal 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 GPACTRL[QUALPRD0] setting and GPIO8 to GPIO15 use GPACTRL[QUALPRD1]. Table 8-1 and Table 8-2 show the relationship between the sampling period or sampling frequency and the GPxCTRL[QUALPRDn] setting.

**Table 8-1. Sampling Period**

| | Sampling Period |
|---|---|
| If GPxCTRL[QUALPRDn] = 0 | $1 \times T_{SYSCLKOUT}$ |
| If GPxCTRL[QUALPRDn] ≠ 0 | $2 \times$ GPxCTRL[QUALPRDn] $\times T_{SYSCLKOUT}$ |
| | Where $T_{SYSCLKOUT}$ is the period in time of SYSCLKOUT |

**Table 8-2. Sampling Frequency**

| | Sampling Frequency |
|---|---|
| If GPxCTRL[QUALPRDn] = 0 | $f_{SYSCLKOUT}$ |
| If GPxCTRL[QUALPRDn] ≠ 0 | $f_{SYSCLKOUT} \times 1 \div (2 \times$ GPxCTRL[QUALPRDn]) |
| | Where $f_{SYSCLKOUT}$ is the frequency of SYSCLKOUT |



Copyright © 2024 Texas Instruments Incorporated

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 $f_{SYSCLKOUT}$

If, for example, $f_{SYSCLKOUT}$ = 60MHz

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 $f_{SYSCLKOUT} \times 1 \div (2 \times$ GPxCTRL[QUALPRDn])

If, for example, $f_{SYSCLKOUT}$ = 60MHz

then the signal is sampled at 60MHz × 1 ÷ (2 × 255) (117.647kHz) or one sample every 8.5µs.

**Number of samples:**

The number of times the signal is sampled is either three samples or six samples as specified in the qualification selection (GPAQSEL1, GPAQSEL2, GPBQSEL1, and GPBQSEL2) registers. When three or six consecutive cycles are the same, then the input change 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 8-1. Likewise, for a six-sample window, the sampling-window width is five sampling-periods wide. Table 8-3 and Table 8-4 show the calculations used to determine the total sampling-window width based on GPxCTRL[QUALPRDn] and the number of samples taken.

**Table 8-3. Case 1: Three-Sample Sampling-Window Width**

| | Total Sampling-Window Width |
|---|---|
| If GPxCTRL[QUALPRDn] = 0 | 2 × $T_{SYSCLKOUT}$ |
| If GPxCTRL[QUALPRDn] ≠ 0 | 2 × 2 × GPxCTRL[QUALPRDn] × $T_{SYSCLKOUT}$ |
| | Where $T_{SYSCLKOUT}$ is the period in time of SYSCLKOUT |

**Table 8-4. Case 2: Six-Sample Sampling-Window Width**

| | Total Sampling-Window Width |
|---|---|
| If GPxCTRL[QUALPRDn] = 0 | 5 × $T_{SYSCLKOUT}$ |
| If GPxCTRL[QUALPRDn] ≠ 0 | 5 × 2 × GPxCTRL[QUALPRDn] × $T_{SYSCLKOUT}$ |
| | Where $T_{SYSCLKOUT}$ is the period in time of SYSCLKOUT |



Copyright © 2024 Texas Instruments Incorporated

**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 + $T_{SYSCLKOUT}$.

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:

- GPxQSEL1/2 = 1,0. This indicates a six-sample qualification.
- GPxCTRL[QUALPRDn] = 1. The sampling period is $t_w(SP) = 2 \times GPxCTRL[QUALPRDn] \times T_{SYSCLKOUT} = 2 \times T_{SYSCLKOUT}$.

This configuration results in the following:

- The width of the sampling window is:

  $t_w(IQSW) = 5 \times t_w(SP) = 5 \times 2 \times GPxCTRL[QUALPRDn] \times T_{SYSCLKOUT} = 5 \times 2 \times T_{SYSCLKOUT}$

- If, for example, $T_{SYSCLKOUT}$ = 16.67ns, then the duration of the sampling window is:

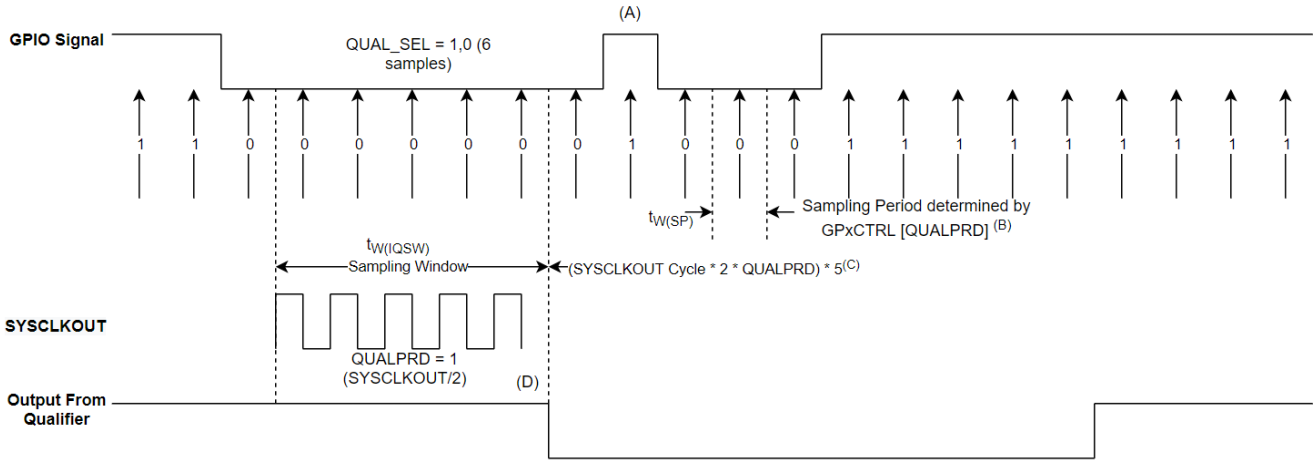
  Sampling period, $t_w(SP) = 2 \times T_{SYSCLKOUT} = 2 \times 16.67ns = 33.3ns$

  Sampling window, $t_w(IQSW) = 5 \times t_w(SP) = 5 \times 33.3ns = 166.7ns$

- 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:

  $t_w(IQSW) + t_w(SP) + T_{SYSCLKOUT} = 166.7ns + 33.3ns + 16.67ns = 216.7ns$

- In Figure 8-3, the glitch (A) is shorter then 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 in 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 x QUALPRD x 2) SYSCLKOUT cycles. That would ensure 5 sampling periods for detection to occur. Since external signals are driven asynchronously, an 13-SYSCLKOUT-wide pulse ensures reliable recognition.

**Figure 8-3. Input Qualifier Clock Cycles**



Copyright © 2024 Texas Instruments Incorporated

## 8.5 USB Signals

The USB module on this device has an internal physical layer transceiver (PHY). The I/O signals are not normal digital signals, and as a result, the signals do not connect to the pins through the normal GPIO mux path. Instead, a special analog mux is used. To connect the USB signals to the device pins, set the GPyAMSEL bits appropriately as shown in Table 8-5. Do not enable pullups or any other special pin option when using the USB signals.

**Table 8-5. USB I/O Signal Muxing**

| Signal | GPIO | AMSEL |
|---|---|---|
| USBDM | 42 | GPBAMSEL[10] |
| USBDP | 43 | GPBAMSEL[11] |

## 8.6 SPI Signals

The SPI module on this device has a high-speed mode that enables 40 Mbps communication. To achieve the highest possible speed, a special GPIO configuration is used on a single GPIO mux option for each SPI. These GPIOs can also be used by the SPI when not in high-speed mode (HS_MODE = 0). Table 8-6 shows which GPIOs have the special mux option to allow SPI high-speed mode.

To select these mux options, configure the GPyGMUX and GPyMUX registers as shown in Table 8-6.

**Table 8-6. GPIO Configuration for High-Speed SPI**

| GPIO | SPI Signal | Mux Configuration | |
|---|---|---|---|
| GPIO58 | SPISIMOA | GPBGMUX2[21:20]=11 | GPBMUX2[21:20]=11 |
| GPIO59 | SPISOMIA | GPBGMUX2[23:22]=11 | GPBMUX2[23:22]=11 |
| GPIO60 | SPICLKA | GPBGMUX2[25:24]=11 | GPBMUX2[25:24]=11 |
| GPIO61 | $\overline{\text{SPISTEA}}$ | GPBGMUX2[27:26]=11 | GPBMUX2[27:26]=11 |
| GPIO63 | SPISIMOB | GPBGMUX2[31:30]=11 | GPBMUX2[31:30]=11 |
| GPIO64 | SPISOMIB | GPCGMUX1[1:0]=11 | GPCMUX1[1:0]=11 |
| GPIO65 | SPICLKB | GPCGMUX1[3:2]=11 | GPCMUX1[3:2]=11 |
| GPIO66 | $\overline{\text{SPISTEB}}$ | GPCGMUX1[5:4]=11 | GPCMUX1[5:4]=11 |
| GPIO69 | SPISIMOC | GPCGMUX1[11:10]=11 | GPCMUX1[11:10]=11 |
| GPIO70 | SPISOMIC | GPCGMUX1[13:12]=11 | GPCMUX1[13:12]=11 |
| GPIO71 | SPICLKC | GPCGMUX1[15:14]=11 | GPCMUX1[15:14]=11 |
| GPIO72 | $\overline{\text{SPISTEC}}$ | GPCGMUX1[17:16]=11 | GPCMUX1[17:16]=11 |



Copyright © 2024 Texas Instruments Incorporated

## 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 8-7 for muxing combinations and definitions.

**Table 8-7. GPIO Muxed Pins**

| | GPIO Mux Selection (1) (2) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **GPIO Index** | **0, 4, 8, 12** | **1** | **2** | **3** | **5** | **6** | **7** | **15** |
| **GPyGMUXn. GPIOz =** | **00b, 01b, 10b, 11b** | **00b** | | | **01b** | | | **11b** |
| **GPyMUXn. GPIOz =** | **00b** | **01b** | **10b** | **11b** | **01b** | **10b** | **11b** | **11b** |
| | GPIO0 | EPWM1A (O) | | | | SDAA (I/OD) | | |
| | GPIO1 | EPWM1B (O) | | MFSRB (I/O) | | SCLA (I/OD) | | |
| | GPIO2 | EPWM2A (O) | | | OUTPUTXBAR1 (O) | SDAB (I/OD) | | |
| | GPIO3 | EPWM2B (O) | OUTPUTXBAR2 (O) | MCLKRB (I/O) | OUTPUTXBAR2 (O) | SCLB (I/OD) | | |
| | GPIO4 | EPWM3A (O) | | | OUTPUTXBAR3 (O) | CANTXA (O) | | |
| | GPIO5 | EPWM3B (O) | MFSRA (I/O) | OUTPUTXBAR3 (O) | | CANRXA (I) | | |
| | GPIO6 | EPWM4A (O) | OUTPUTXBAR4 (O) | EXTSYNCOUT (O) | EQEP3A (I) | CANTXB (O) | | |
| | GPIO7 | EPWM4B (O) | MCLKRA (I/O) | OUTPUTXBAR5 (O) | EQEP3B (I) | CANRXB (I) | | |
| | GPIO8 | EPWM5A (O) | CANTXB (O) | ADCSOCAO (O) | EQEP3S (I/O) | SCITXDA (O) | | |
| | GPIO9 | EPWM5B (O) | SCITXDB (O) | OUTPUTXBAR6 (O) | EQEP3I (I/O) | SCIRXDA (I) | | |
| | GPIO10 | EPWM6A (O) | CANRXB (I) | ADCSOCBO (O) | EQEP1A (I) | SCITXDB (O) | | UPP-WAIT (I/O) |
| | GPIO11 | EPWM6B (O) | SCIRXDB (I) | OUTPUTXBAR7 (O) | EQEP1B (I) | SCIRXDB (I) | | UPP-START (I/O) |
| | GPIO12 | EPWM7A (O) | CANTXB (O) | MDXB (O) | EQEP1S (I/O) | SCITXDC (O) | | UPP-ENA (I/O) |
| | GPIO13 | EPWM7B (O) | CANRXB (I) | MDRB (I) | EQEP1I (I/O) | SCIRXDC (I) | | UPP-D7 (I/O) |
| | GPIO14 | EPWM8A (O) | SCITXDB (O) | MCLKXB (I/O) | | OUTPUTXBAR3 (O) | | UPP-D6 (I/O) |
| | GPIO15 | EPWM8B (O) | SCIRXDB (I) | MFSXB (I/O) | | OUTPUTXBAR4 (O) | | UPP-D5 (I/O) |
| | GPIO16 | SPISIMOA (I/O) | CANTXB (O) | OUTPUTXBAR7 (O) | EPWM9A (O) | | SD1_D1 (I) | UPP-D4 (I/O) |
| | GPIO17 | SPISOMIA (I/O) | CANRXB (I) | OUTPUTXBAR8 (O) | EPWM9B (O) | | SD1_C1 (I) | UPP-D3 (I/O) |
| | GPIO18 | SPICLKA (I/O) | SCITXDB (O) | CANRXA (I) | EPWM10A (O) | | SD1_D2 (I) | UPP-D2 (I/O) |
| | GPIO19 | $\overline{\text{SPISTEA}}$ (I/O) | SCIRXDB (I) | CANTXA (O) | EPWM10B (O) | | SD1_C2 (I) | UPP-D1 (I/O) |
| | GPIO20 | EQEP1A (I) | MDXA (O) | CANTXB (O) | EPWM11A (O) | | SD1_D3 (I) | UPP-D0 (I/O) |
| | GPIO21 | EQEP1B (I) | MDRA (I) | CANRXB (I) | EPWM11B (O) | | SD1_C3 (I) | UPP-CLK (I/O) |
| | GPIO22 | EQEP1S (I/O) | MCLKXA (I/O) | SCITXDB (O) | EPWM12A (O) | SPICLKB (I/O) | SD1_D4 (I) | |
| | GPIO23 | EQEP1I (I/O) | MFSXA (I/O) | SCIRXDB (I) | EPWM12B (O) | $\overline{\text{SPISTEB}}$ (I/O) | SD1_C4 (I) | |
| | GPIO24 | OUTPUTXBAR1 (O) | EQEP2A (I) | MDXB (O) | | SPISIMOB (I/O) | SD2_D1 (I) | |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-7. GPIO Muxed Pins (continued)**

| | GPIO Mux Selection [1] [2] | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **GPIO Index** | **0, 4, 8, 12** | **1** | **2** | **3** | **5** | **6** | **7** | **15** |
| **GPyGMUXn. GPIOz =** | **00b, 01b, 10b, 11b** | **00b** | | | **01b** | | | **11b** |
| **GPyMUXn. GPIOz =** | **00b** | **01b** | **10b** | **11b** | **01b** | **10b** | **11b** | **11b** |
| | GPIO25 | OUTPUTXBAR2 (O) | EQEP2B (I) | MDRB (I) | | SPISOMIB (I/O) | SD2_C1 (I) | |
| | GPIO26 | OUTPUTXBAR3 (O) | EQEP2I (I/O) | MCLKXB (I/O) | OUTPUTXBAR3 (O) | SPICLKB (I/O) | SD2_D2 (I) | |
| | GPIO27 | OUTPUTXBAR4 (O) | EQEP2S (I/O) | MFSXB (I/O) | OUTPUTXBAR4 (O) | $\overline{\text{SPISTEB}}$ (I/O) | SD2_C2 (I) | |
| | GPIO28 | SCIRXDA (I) | $\overline{\text{EM1CS4}}$ (O) | | OUTPUTXBAR5 (O) | EQEP3A (I) | SD2_D3 (I) | |
| | GPIO29 | SCITXDA (O) | EM1SDCKE (O) | | OUTPUTXBAR6 (O) | EQEP3B (I) | SD2_C3 (I) | |
| | GPIO30 | CANRXA (I) | EM1CLK (O) | | OUTPUTXBAR7 (O) | EQEP3S (I/O) | SD2_D4 (I) | |
| | GPIO31 | CANTXA (O) | $\overline{\text{EM1WE}}$ (O) | | OUTPUTXBAR8 (O) | EQEP3I (I/O) | SD2_C4 (I) | |
| | GPIO32 | SDAA (I/OD) | $\overline{\text{EM1CS0}}$ (O) | | | | | |
| | GPIO33 | SCLA (I/OD) | EM1RNW (O) | | | | | |
| | GPIO34 | OUTPUTXBAR1 (O) | $\overline{\text{EM1CS2}}$ (O) | | | SDAB (I/OD) | | |
| | GPIO35 | SCIRXDA (I) | $\overline{\text{EM1CS3}}$ (O) | | | SCLB (I/OD) | | |
| | GPIO36 | SCITXDA (O) | EM1WAIT (I) | | | CANRXA (I) | | |
| | GPIO37 | OUTPUTXBAR2 (O) | $\overline{\text{EM1OE}}$ (O) | | | CANTXA (O) | | |
| | GPIO38 | | EM1A0 (O) | | SCITXDC (O) | CANTXB (O) | | |
| | GPIO39 | | EM1A1 (O) | | SCIRXDC (I) | CANRXB (I) | | |
| | GPIO40 | | EM1A2 (O) | | | SDAB (I/OD) | | |
| | GPIO41 | | EM1A3 (O) | | | SCLB (I/OD) | | |
| | GPIO42 | | | | | SDAA (I/OD) | | SCITXDA (O) |
| | GPIO43 | | | | | SCLA (I/OD) | | SCIRXDA (I) |
| | GPIO44 | | EM1A4 (O) | | | | | |
| | GPIO45 | | EM1A5 (O) | | | | | |
| | GPIO46 | | EM1A6 (O) | | | SCIRXDD (I) | | |
| | GPIO47 | | EM1A7 (O) | | | SCITXDD (O) | | |
| | GPIO48 | OUTPUTXBAR3 (O) | EM1A8 (O) | | | SCITXDA (O) | SD1_D1 (I) | |
| | GPIO49 | OUTPUTXBAR4 (O) | EM1A9 (O) | | | SCIRXDA (I) | SD1_C1 (I) | |
| | GPIO50 | EQEP1A (I) | EM1A10 (O) | | | SPISIMOC (I/O) | SD1_D2 (I) | |
| | GPIO51 | EQEP1B (I) | EM1A11 (O) | | | SPISOMIC (I/O) | SD1_C2 (I) | |
| | GPIO52 | EQEP1S (I/O) | EM1A12 (O) | | | SPICLKC (I/O) | SD1_D3 (I) | |
| | GPIO53 | EQEP1I (I/O) | EM1D31 (I/O) | EM2D15 (I/O) | | $\overline{\text{SPISTEC}}$ (I/O) | SD1_C3 (I) | |
| | GPIO54 | SPISIMOA (I/O) | EM1D30 (I/O) | EM2D14 (I/O) | EQEP2A (I) | SCITXDB (O) | SD1_D4 (I) | |
| | GPIO55 | SPISOMIA (I/O) | EM1D29 (I/O) | EM2D13 (I/O) | EQEP2B (I) | SCIRXDB (I) | SD1_C4 (I) | |



Copyright © 2024 Texas Instruments Incorporated

## Table 8-7. GPIO Muxed Pins (continued)

| | GPIO Mux Selection [1] [2] | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **GPIO Index** | **0, 4, 8, 12** | **1** | **2** | **3** | **5** | **6** | **7** | **15** |
| **GPyGMUXn. GPIOz =** | **00b, 01b, 10b, 11b** | **00b** | | | **01b** | | | **11b** |
| **GPyMUXn. GPIOz =** | **00b** | **01b** | **10b** | **11b** | **01b** | **10b** | **11b** | **11b** |
| | GPIO56 | SPICLKA (I/O) | EM1D28 (I/O) | EM2D12 (I/O) | EQEP2S (I/O) | SCITXDC (O) | SD2_D1 (I) | |
| | GPIO57 | $\overline{\text{SPISTEA}}$ (I/O) | EM1D27 (I/O) | EM2D11 (I/O) | EQEP2I (I/O) | SCIRXDC (I) | SD2_C1 (I) | |
| | GPIO58 | MCLKRA (I/O) | EM1D26 (I/O) | EM2D10 (I/O) | OUTPUTXBAR1 (O) | SPICLKB (I/O) | SD2_D2 (I) | SPISIMOA[3] (I/O) |
| | GPIO59 | MFSRA (I/O) | EM1D25 (I/O) | EM2D9 (I/O) | OUTPUTXBAR2 (O) | $\overline{\text{SPISTEB}}$ (I/O) | SD2_C2 (I) | SPISOMIA[3] (I/O) |
| | GPIO60 | MCLKRB (I/O) | EM1D24 (I/O) | EM2D8 (I/O) | OUTPUTXBAR3 (O) | SPISIMOB (I/O) | SD2_D3 (I) | SPICLKA[3] (I/O) |
| | GPIO61 | MFSRB (I/O) | EM1D23 (I/O) | EM2D7 (I/O) | OUTPUTXBAR4 (O) | SPISOMIB (I/O) | SD2_C3 (I) | $\overline{\text{SPISTEA}}$ [3] (I/O) |
| | GPIO62 | SCIRXDC (I) | EM1D22 (I/O) | EM2D6 (I/O) | EQEP3A (I) | CANRXA (I) | SD2_D4 (I) | |
| | GPIO63 | SCITXDC (O) | EM1D21 (I/O) | EM2D5 (I/O) | EQEP3B (I) | CANTXA (O) | SD2_C4 (I) | SPISIMOB[3] (I/O) |
| | GPIO64 | | EM1D20 (I/O) | EM2D4 (I/O) | EQEP3S (I/O) | SCIRXDA (I) | | SPISOMIB[3] (I/O) |
| | GPIO65 | | EM1D19 (I/O) | EM2D3 (I/O) | EQEP3I (I/O) | SCITXDA (O) | | SPICLKB[3] (I/O) |
| | GPIO66 | | EM1D18 (I/O) | EM2D2 (I/O) | | SDAB (I/OD) | | $\overline{\text{SPISTEB}}$ [3] (I/O) |
| | GPIO67 | | EM1D17 (I/O) | EM2D1 (I/O) | | | | |
| | GPIO68 | | EM1D16 (I/O) | EM2D0 (I/O) | | | | |
| | GPIO69 | | EM1D15 (I/O) | | | SCLB (I/OD) | | SPISIMOC[3] (I/O) |
| | GPIO70 | | EM1D14 (I/O) | | CANRXA (I) | SCITXDB (O) | | SPISOMIC[3] (I/O) |
| | GPIO71 | | EM1D13 (I/O) | | CANTXA (O) | SCIRXDB (I) | | SPICLKC[3] (I/O) |
| | GPIO72 | | EM1D12 (I/O) | | CANTXB (O) | SCITXDC (O) | | $\overline{\text{SPISTEC}}$ [3] (I/O) |
| | GPIO73 | | EM1D11 (I/O) | XCLKOUT (O) | CANRXB (I) | SCIRXDC (I) | | |
| | GPIO74 | | EM1D10 (I/O) | | | | | |
| | GPIO75 | | EM1D9 (I/O) | | | | | |
| | GPIO76 | | EM1D8 (I/O) | | | SCITXDD (O) | | |
| | GPIO77 | | EM1D7 (I/O) | | | SCIRXDD (I) | | |
| | GPIO78 | | EM1D6 (I/O) | | | EQEP2A (I) | | |
| | GPIO79 | | EM1D5 (I/O) | | | EQEP2B (I) | | |
| | GPIO80 | | EM1D4 (I/O) | | | EQEP2S (I/O) | | |
| | GPIO81 | | EM1D3 (I/O) | | | EQEP2I (I/O) | | |
| | GPIO82 | | EM1D2 (I/O) | | | | | |
| | GPIO83 | | EM1D1 (I/O) | | | | | |
| | GPIO84 | | | | SCITXDA (O) | MDXB (O) | | MDXA (O) |
| | GPIO85 | | EM1D0 (I/O) | | SCIRXDA (I) | MDRB (I) | | MDRA (I) |
| | GPIO86 | | EM1A13 (O) | EM1CAS (O) | SCITXDB (O) | MCLKXB (I/O) | | MCLKXA (I/O) |



Copyright © 2024 Texas Instruments Incorporated

## Table 8-7. GPIO Muxed Pins (continued)

| | GPIO Mux Selection [1] [2] | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **GPIO Index** | **0, 4, 8, 12** | **1** | **2** | **3** | **5** | **6** | **7** | **15** |
| **GPyGMUXn. GPIOz =** | **00b, 01b, 10b, 11b** | **00b** | | | **01b** | | | **11b** |
| **GPyMUXn. GPIOz =** | **00b** | **01b** | **10b** | **11b** | **01b** | **10b** | **11b** | **11b** |
| | GPIO87 | | EM1A14 (O) | EM1RAS (O) | SCIRXDB (I) | MFSXB (I/O) | | MFSXA (I/O) |
| | GPIO88 | | EM1A15 (O) | EM1DQM0 (O) | | | | |
| | GPIO89 | | EM1A16 (O) | EM1DQM1 (O) | | SCITXDC (O) | | |
| | GPIO90 | | EM1A17 (O) | EM1DQM2 (O) | | SCIRXDC (I) | | |
| | GPIO91 | | EM1A18 (O) | EM1DQM3 (O) | | SDAA (I/OD) | | |
| | GPIO92 | | EM1A19 (O) | EM1BA1 (O) | | SCLA (I/OD) | | |
| | GPIO93 | | | EM1BA0 (O) | | SCITXDD (O) | | |
| | GPIO94 | | | | | SCIRXDD (I) | | |
| | GPIO95 | | | | | | | |
| | GPIO96 | | | EM2DQM1 (O) | EQEP1A (I) | | | |
| | GPIO97 | | | EM2DQM0 (O) | EQEP1B (I) | | | |
| | GPIO98 | | | EM2A0 (O) | EQEP1S (I/O) | | | |
| | GPIO99 | | | EM2A1 (O) | EQEP1I (I/O) | | | |
| | GPIO100 | | | EM2A2 (O) | EQEP2A (I) | SPISIMOC (I/O) | | |
| | GPIO101 | | | EM2A3 (O) | EQEP2B (I) | SPISOMIC (I/O) | | |
| | GPIO102 | | | EM2A4 (O) | EQEP2S (I/O) | SPICLKC (I/O) | | |
| | GPIO103 | | | EM2A5 (O) | EQEP2I (I/O) | $\overline{\text{SPISTEC}}$ (I/O) | | |
| | GPIO104 | SDAA (I/OD) | | EM2A6 (O) | EQEP3A (I) | SCITXDD (O) | | |
| | GPIO105 | SCLA (I/OD) | | EM2A7 (O) | EQEP3B (I) | SCIRXDD (I) | | |
| | GPIO106 | | | EM2A8 (O) | EQEP3S (I/O) | SCITXDC (O) | | |
| | GPIO107 | | | EM2A9 (O) | EQEP3I (I/O) | SCIRXDC (I) | | |
| | GPIO108 | | | EM2A10 (O) | | | | |
| | GPIO109 | | | EM2A11 (O) | | | | |
| | GPIO110 | | | EM2WAIT (I) | | | | |
| | GPIO111 | | | EM2BA0 (O) | | | | |
| | GPIO112 | | | EM2BA1 (O) | | | | |
| | GPIO113 | | | EM2CAS (O) | | | | |
| | GPIO114 | | | EM2RAS (O) | | | | |
| | GPIO115 | | | $\overline{\text{EM2CS0}}$ (O) | | | | |
| | GPIO116 | | | $\overline{\text{EM2CS2}}$ (O) | | | | |
| | GPIO117 | | | EM2SDCKE (O) | | | | |



Copyright © 2024 Texas Instruments Incorporated

## Table 8-7. GPIO Muxed Pins (continued)

| | GPIO Mux Selection (1) (2) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **GPIO Index** | **0, 4, 8, 12** | **1** | **2** | **3** | **5** | **6** | **7** | **15** |
| **GPyGMUXn. GPIOz =** | **00b, 01b, 10b, 11b** | **00b** | | | **01b** | | | **11b** |
| **GPyMUXn. GPIOz =** | **00b** | **01b** | **10b** | **11b** | **01b** | **10b** | **11b** | **11b** |
| | GPIO118 | | | EM2CLK (O) | | | | |
| | GPIO119 | | | EM2RNW (O) | | | | |
| | GPIO120 | | | $\overline{\text{EM2WE}}$ (O) | | | | USB0PFLT |
| | GPIO121 | | | $\overline{\text{EM2OE}}$ (O) | | | | USB0EPEN |
| | GPIO122 | | | | | SPISIMOC (I/O) | SD1_D1 (I) | |
| | GPIO123 | | | | | SPISOMIC (I/O) | SD1_C1 (I) | |
| | GPIO124 | | | | | SPICLKC (I/O) | SD1_D2 (I) | |
| | GPIO125 | | | | | $\overline{\text{SPISTEC}}$ (I/O) | SD1_C2 (I) | |
| | GPIO126 | | | | | | SD1_D3 (I) | |
| | GPIO127 | | | | | | SD1_C3 (I) | |
| | GPIO128 | | | | | | SD1_D4 (I) | |
| | GPIO129 | | | | | | SD1_C4 (I) | |
| | GPIO130 | | | | | | SD2_D1 (I) | |
| | GPIO131 | | | | | | SD2_C1 (I) | |
| | GPIO132 | | | | | | SD2_D2 (I) | |
| | GPIO133/ AUXCLKIN | | | | | | SD2_C2 (I) | |
| | GPIO134 | | | | | | SD2_D3 (I) | |
| | GPIO135 | | | | | SCITXDA (O) | SD2_C3 (I) | |
| | GPIO136 | | | | | SCIRXDA (I) | SD2_D4 (I) | |
| | GPIO137 | | | | | SCITXDB (O) | SD2_C4 (I) | |
| | GPIO138 | | | | | SCIRXDB (I) | | |
| | GPIO139 | | | | | SCIRXDC (I) | | |
| | GPIO140 | | | | | SCITXDC (O) | | |
| | GPIO141 | | | | | SCIRXDD (I) | | |
| | GPIO142 | | | | | SCITXDD (O) | | |
| | GPIO143 | | | | | | | |
| | GPIO144 | | | | | | | |
| | GPIO145 | EPWM1A (O) | | | | | | |
| | GPIO146 | EPWM1B (O) | | | | | | |
| | GPIO147 | EPWM2A (O) | | | | | | |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-7. GPIO Muxed Pins (continued)**

| | GPIO Mux Selection (1) (2) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **GPIO Index** | **0, 4, 8, 12** | **1** | **2** | **3** | **5** | **6** | **7** | **15** |
| **GPyGMUXn. GPIOz =** | **00b, 01b, 10b, 11b** | **00b** | | | **01b** | | | **11b** |
| **GPyMUXn. GPIOz =** | **00b** | **01b** | **10b** | **11b** | **01b** | **10b** | **11b** | **11b** |
| | GPIO148 | EPWM2B (O) | | | | | | |
| | GPIO149 | EPWM3A (O) | | | | | | |
| | GPIO150 | EPWM3B (O) | | | | | | |
| | GPIO151 | EPWM4A (O) | | | | | | |
| | GPIO152 | EPWM4B (O) | | | | | | |
| | GPIO153 | EPWM5A (O) | | | | | | |
| | GPIO154 | EPWM5B (O) | | | | | | |
| | GPIO155 | EPWM6A (O) | | | | | | |
| | GPIO156 | EPWM6B (O) | | | | | | |
| | GPIO157 | EPWM7A (O) | | | | | | |
| | GPIO158 | EPWM7B (O) | | | | | | |
| | GPIO159 | EPWM8A (O) | | | | | | |
| | GPIO160 | EPWM8B (O) | | | | | | |
| | GPIO161 | EPWM9A (O) | | | | | | |
| | GPIO162 | EPWM9B (O) | | | | | | |
| | GPIO163 | EPWM10A (O) | | | | | | |
| | GPIO164 | EPWM10B (O) | | | | | | |
| | GPIO165 | EPWM11A (O) | | | | | | |
| | GPIO166 | EPWM11B (O) | | | | | | |
| | GPIO167 | EPWM12A (O) | | | | | | |
| | GPIO168 | EPWM12B (O) | | | | | | |

(1) (I) = Input, (O) = Output, (OD) = Open Drain
(2) GPIO Index settings of 9, 10, 11, 13, and 14 are reserved.
(3) High-Speed SPI-enabled GPIO mux option. This pin mux option is required when using the SPI in High-Speed Mode (HS_MODE = 1 in SPICCR). This mux option is still available when not using the SPI in High-Speed Mode (HS_MODE = 0 in SPICCR).



Copyright © 2024 Texas Instruments Incorporated

### 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 8-8.

**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 8-8. 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 8-9.

**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.



Copyright © 2024 Texas Instruments Incorporated

If none or more then one of the GPIO pins is configured as peripheral input pins, then that GPIO is set to a hard-wired default value.

**Table 8-9. 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 |



Copyright © 2024 Texas Instruments Incorporated

## 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.



Copyright © 2024 Texas Instruments Incorporated

## 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.

#### 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 (CAN, 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 swtich button under the device view and select your corresponding device to migrate, saving the project will auto-migrate your project settings.

: This example project has support for migration across our C2000 device families. If you are wanting to build this project from launchpad or controlCARD, please specify in the .syscfg file the board you're using. At any time you can select another device to migrate this example.

#### 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.

### *8.9.2 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 C2000Ware Examples.



Copyright © 2024 Texas Instruments Incorporated

## 8.10 GPIO Registers

This section describes the General-Purpose Input/Output Registers.

Table 8-10 provides both specific and generic terms for registers described in this section and throughout this chapter.

**Table 8-10. Specific versus Generic Terminology for Registers**

| Specific Term | Generic Term Used in the Document |
|---|---|
| GPAQSEL1, GPAQSEL2, GPBQSEL1, GPBQSEL2, GPCQSEL1, GPCQSEL2, GPDQSEL1, GPDQSEL2, GPEQSEL1, GPEQSEL2, GPFQSEL1, GPFQSEL2 | GPxQSEL |
| GPACTRL, GPBCTRL, GPCCTRL, GPDCTRL, GPECTRL, GPFCTRL | GPxCTRL |
| GPADIR, GPBDIR, GPCDIR, GPDDIR, GPEDIR, GPFDIR | GPxDIR |
| GPAPUD, GPBPUD, GPCPUD, GPDPUD, GPEPUD, GPFPUD | GPxPUD |
| GPAINV, GPBINV, GPCINV, GPDINV, GPEINV, GPFINV | GPxINV |
| GPAODR, GPBODR, GPCODR, GPDODR, GPEODR, GPFODR, GPxODR | GPxODR |
| GPALOCK, GPBLOCK, GPCLOCK, GPDLOCK, GPELOCK, GPFLOCK | GPxLOCK |
| GPACR, GPBCR, GPCCR, GPDCR, GPECR, GPFCR | GPxCR |
| GPAMUX1, GPAMUX2, GPBMUX1, GPBMUX2, GPCMUX1, GPCMUX2, GPDMUX1, GPDMUX2, GPEMUX1, GPEMUX2, GPFMUX1, GPFMUX2 | GPxMUX |
| GPAGMUX1, GPAGMUX2, GPBGMUX1, GPBGMUX2, GPCGMUX1, GPCGMUX2, GPDGMUX1, GPDGMUX2, GPEGMUX1, GPEGMUX2, GPFGMUX1, GPFGMUX2 | GPxGMUX |
| GPADAT, GPBDAT, GPCDAT, GPDDAT, GPEDAT, GPFDAT | GPxDAT |

### *8.10.1 GPIO Base Addresses*

**Table 8-11. GPIO Base Address Table**

| Device Registers | Register Name | Start Address | End Address |
|---|---|---|---|
| GpioCtrlRegs(1) | GPIO_CTRL_REGS | 0x0000_7C00 | 0x0000_7D7F |
| GpioDataRegs | GPIO_DATA_REGS | 0x0000_7F00 | 0x0000_7F2F |

(1) Only available on CPU1.



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2 GPIO_CTRL_REGS Registers

Table 8-12 lists the memory-mapped registers for the GPIO_CTRL_REGS registers. All register offset addresses not listed in Table 8-12 should be considered as reserved locations and the register contents should not be modified.

**Table 8-12. GPIO_CTRL_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | GPACTRL | GPIO A Qualification Sampling Period Control (GPIO0 to 31) | EALLOW | Go |
| 2h | GPAQSEL1 | GPIO A Qualifier Select 1 Register (GPIO0 to 15) | EALLOW | Go |
| 4h | GPAQSEL2 | GPIO A Qualifier Select 2 Register (GPIO16 to 31) | EALLOW | Go |
| 6h | GPAMUX1 | GPIO A Mux 1 Register (GPIO0 to 15) | EALLOW | Go |
| 8h | GPAMUX2 | GPIO A Mux 2 Register (GPIO16 to 31) | EALLOW | Go |
| Ah | GPADIR | GPIO A Direction Register (GPIO0 to 31) | EALLOW | Go |
| Ch | GPAPUD | GPIO A Pull Up Disable Register (GPIO0 to 31) | EALLOW | Go |
| 10h | GPAINV | GPIO A Input Polarity Invert Registers (GPIO0 to 31) | EALLOW | Go |
| 12h | GPAODR | GPIO A Open Drain Output Register (GPIO0 to GPIO31) | EALLOW | Go |
| 20h | GPAGMUX1 | GPIO A Peripheral Group Mux (GPIO0 to 15) | EALLOW | Go |
| 22h | GPAGMUX2 | GPIO A Peripheral Group Mux (GPIO16 to 31) | EALLOW | Go |
| 28h | GPACSEL1 | GPIO A Core Select Register (GPIO0 to 7) | EALLOW | Go |
| 2Ah | GPACSEL2 | GPIO A Core Select Register (GPIO8 to 15) | EALLOW | Go |
| 2Ch | GPACSEL3 | GPIO A Core Select Register (GPIO16 to 23) | EALLOW | Go |
| 2Eh | GPACSEL4 | GPIO A Core Select Register (GPIO24 to 31) | EALLOW | Go |
| 3Ch | GPALOCK | GPIO A Lock Configuration Register (GPIO0 to 31) | EALLOW | Go |
| 3Eh | GPACR | GPIO A Lock Commit Register (GPIO0 to 31) | EALLOW | Go |
| 40h | GPBCTRL | GPIO B Qualification Sampling Period Control (GPIO32 to 63) | EALLOW | Go |
| 42h | GPBQSEL1 | GPIO B Qualifier Select 1 Register (GPIO32 to 47) | EALLOW | Go |
| 44h | GPBQSEL2 | GPIO B Qualifier Select 2 Register (GPIO48 to 63) | EALLOW | Go |
| 46h | GPBMUX1 | GPIO B Mux 1 Register (GPIO32 to 47) | EALLOW | Go |
| 48h | GPBMUX2 | GPIO B Mux 2 Register (GPIO48 to 63) | EALLOW | Go |
| 4Ah | GPBDIR | GPIO B Direction Register (GPIO32 to 63) | EALLOW | Go |
| 4Ch | GPBPUD | GPIO B Pull Up Disable Register (GPIO32 to 63) | EALLOW | Go |
| 50h | GPBINV | GPIO B Input Polarity Invert Registers (GPIO32 to 63) | EALLOW | Go |
| 52h | GPBODR | GPIO B Open Drain Output Register (GPIO32 to GPIO63) | EALLOW | Go |
| 54h | GPBAMSEL | GPIO B Analog Mode Select register (GPIO32 to GPIO63) | EALLOW | Go |
| 60h | GPBGMUX1 | GPIO B Peripheral Group Mux (GPIO32 to 47) | EALLOW | Go |
| 62h | GPBGMUX2 | GPIO B Peripheral Group Mux (GPIO48 to 63) | EALLOW | Go |
| 68h | GPBCSEL1 | GPIO B Core Select Register (GPIO32 to 39) | EALLOW | Go |
| 6Ah | GPBCSEL2 | GPIO B Core Select Register (GPIO40 to 47) | EALLOW | Go |
| 6Ch | GPBCSEL3 | GPIO B Core Select Register (GPIO48 to 55) | EALLOW | Go |
| 6Eh | GPBCSEL4 | GPIO B Core Select Register (GPIO56 to 63) | EALLOW | Go |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-12. GPIO_CTRL_REGS Registers (continued)**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 7Ch | GPBLOCK | GPIO B Lock Configuration Register (GPIO32 to 63) | EALLOW | Go |
| 7Eh | GPBCR | GPIO B Lock Commit Register (GPIO32 to 63) | EALLOW | Go |
| 80h | GPCCTRL | GPIO C Qualification Sampling Period Control (GPIO64 to 95) | EALLOW | Go |
| 82h | GPCQSEL1 | GPIO C Qualifier Select 1 Register (GPIO64 to 79) | EALLOW | Go |
| 84h | GPCQSEL2 | GPIO C Qualifier Select 2 Register (GPIO80 to 95) | EALLOW | Go |
| 86h | GPCMUX1 | GPIO C Mux 1 Register (GPIO64 to 79) | EALLOW | Go |
| 88h | GPCMUX2 | GPIO C Mux 2 Register (GPIO80 to 95) | EALLOW | Go |
| 8Ah | GPCDIR | GPIO C Direction Register (GPIO64 to 95) | EALLOW | Go |
| 8Ch | GPCPUD | GPIO C Pull Up Disable Register (GPIO64 to 95) | EALLOW | Go |
| 90h | GPCINV | GPIO C Input Polarity Invert Registers (GPIO64 to 95) | EALLOW | Go |
| 92h | GPCODR | GPIO C Open Drain Output Register (GPIO64 to GPIO95) | EALLOW | Go |
| A0h | GPCGMUX1 | GPIO C Peripheral Group Mux (GPIO64 to 79) | EALLOW | Go |
| A2h | GPCGMUX2 | GPIO C Peripheral Group Mux (GPIO80 to 95) | EALLOW | Go |
| A8h | GPCCSEL1 | GPIO C Core Select Register (GPIO64 to 71) | EALLOW | Go |
| AAh | GPCCSEL2 | GPIO C Core Select Register (GPIO72 to 79) | EALLOW | Go |
| ACh | GPCCSEL3 | GPIO C Core Select Register (GPIO80 to 87) | EALLOW | Go |
| AEh | GPCCSEL4 | GPIO C Core Select Register (GPIO88 to 95) | EALLOW | Go |
| BCh | GPCLOCK | GPIO C Lock Configuration Register (GPIO64 to 95) | EALLOW | Go |
| BEh | GPCCR | GPIO C Lock Commit Register (GPIO64 to 95) | EALLOW | Go |
| C0h | GPDCTRL | GPIO D Qualification Sampling Period Control (GPIO96 to 127) | EALLOW | Go |
| C2h | GPDQSEL1 | GPIO D Qualifier Select 1 Register (GPIO96 to 111) | EALLOW | Go |
| C4h | GPDQSEL2 | GPIO D Qualifier Select 2 Register (GPIO112 to 127) | EALLOW | Go |
| C6h | GPDMUX1 | GPIO D Mux 1 Register (GPIO96 to 111) | EALLOW | Go |
| C8h | GPDMUX2 | GPIO D Mux 2 Register (GPIO112 to 127) | EALLOW | Go |
| CAh | GPDDIR | GPIO D Direction Register (GPIO96 to 127) | EALLOW | Go |
| CCh | GPDPUD | GPIO D Pull Up Disable Register (GPIO96 to 127) | EALLOW | Go |
| D0h | GPDINV | GPIO D Input Polarity Invert Registers (GPIO96 to 127) | EALLOW | Go |
| D2h | GPDODR | GPIO D Open Drain Output Register (GPIO96 to GPIO127) | EALLOW | Go |
| E0h | GPDGMUX1 | GPIO D Peripheral Group Mux (GPIO96 to 111) | EALLOW | Go |
| E2h | GPDGMUX2 | GPIO D Peripheral Group Mux (GPIO112 to 127) | EALLOW | Go |
| E8h | GPDCSEL1 | GPIO D Core Select Register (GPIO96 to 103) | EALLOW | Go |
| EAh | GPDCSEL2 | GPIO D Core Select Register (GPIO104 to 111) | EALLOW | Go |
| ECh | GPDCSEL3 | GPIO D Core Select Register (GPIO112 to 119) | EALLOW | Go |
| EEh | GPDCSEL4 | GPIO D Core Select Register (GPIO120 to 127) | EALLOW | Go |
| FCh | GPDLOCK | GPIO D Lock Configuration Register (GPIO96 to 127) | EALLOW | Go |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-12. GPIO_CTRL_REGS Registers (continued)**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| FEh | GPDCR | GPIO D Lock Commit Register (GPIO96 to 127) | EALLOW | Go |
| 100h | GPECTRL | GPIO E Qualification Sampling Period Control (GPIO128 to 159) | EALLOW | Go |
| 102h | GPEQSEL1 | GPIO E Qualifier Select 1 Register (GPIO128 to 143) | EALLOW | Go |
| 104h | GPEQSEL2 | GPIO E Qualifier Select 2 Register (GPIO144 to 159) | EALLOW | Go |
| 106h | GPEMUX1 | GPIO E Mux 1 Register (GPIO128 to 143) | EALLOW | Go |
| 108h | GPEMUX2 | GPIO E Mux 2 Register (GPIO144 to 159) | EALLOW | Go |
| 10Ah | GPEDIR | GPIO E Direction Register (GPIO128 to 159) | EALLOW | Go |
| 10Ch | GPEPUD | GPIO E Pull Up Disable Register (GPIO128 to 159) | EALLOW | Go |
| 110h | GPEINV | GPIO E Input Polarity Invert Registers (GPIO128 to 159) | EALLOW | Go |
| 112h | GPEODR | GPIO E Open Drain Output Register (GPIO128 to GPIO159) | EALLOW | Go |
| 120h | GPEGMUX1 | GPIO E Peripheral Group Mux (GPIO128 to 143) | EALLOW | Go |
| 122h | GPEGMUX2 | GPIO E Peripheral Group Mux (GPIO144 to 159) | EALLOW | Go |
| 128h | GPECSEL1 | GPIO E Core Select Register (GPIO128 to 135) | EALLOW | Go |
| 12Ah | GPECSEL2 | GPIO E Core Select Register (GPIO136 to 143) | EALLOW | Go |
| 12Ch | GPECSEL3 | GPIO E Core Select Register (GPIO144 to 151) | EALLOW | Go |
| 12Eh | GPECSEL4 | GPIO E Core Select Register (GPIO152 to 159) | EALLOW | Go |
| 13Ch | GPELOCK | GPIO E Lock Configuration Register (GPIO128 to 159) | EALLOW | Go |
| 13Eh | GPECR | GPIO E Lock Commit Register (GPIO128 to 159) | EALLOW | Go |
| 140h | GPFCTRL | GPIO F Qualification Sampling Period Control (GPIO160 to 168) | EALLOW | Go |
| 142h | GPFQSEL1 | GPIO F Qualifier Select 1 Register (GPIO160 to 168) | EALLOW | Go |
| 146h | GPFMUX1 | GPIO F Mux 1 Register (GPIO160 to 168) | EALLOW | Go |
| 14Ah | GPFDIR | GPIO F Direction Register (GPIO160 to 168) | EALLOW | Go |
| 14Ch | GPFPUD | GPIO F Pull Up Disable Register (GPIO160 to 168) | EALLOW | Go |
| 150h | GPFINV | GPIO F Input Polarity Invert Registers (GPIO160 to 168) | EALLOW | Go |
| 152h | GPFODR | GPIO F Open Drain Output Register (GPIO160 to GPIO168) | EALLOW | Go |
| 160h | GPFGMUX1 | GPIO F Peripheral Group Mux (GPIO160 to 168) | EALLOW | Go |
| 168h | GPFCSEL1 | GPIO F Core Select Register (GPIO160 to 167) | EALLOW | Go |
| 16Ah | GPFCSEL2 | GPIO F Core Select Register (GPIO168) | EALLOW | Go |
| 17Ch | GPFLOCK | GPIO F Lock Configuration Register (GPIO160 to 168) | EALLOW | Go |
| 17Eh | GPFCR | GPIO F Lock Commit Register (GPIO160 to 168) | EALLOW | Go |

Complex bit access types are encoded to fit into small table cells. Table 8-13 shows the codes that are used for access types in this section.

**Table 8-13. GPIO_CTRL_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-13. GPIO_CTRL_REGS Access Type Codes (continued)**

| Access Type | Code | Description |
|---|---|---|
| R | R | Read |
| Write Type | | |
| W | W | Write |
| WOnce | W<br>Once | Write<br>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. |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.1 GPACTRL Register (Offset = 0h) [Reset = 00000000h]

GPACTRL is shown in Figure 8-4 and described in Table 8-14.

Return to the Summary Table.

GPIO A Qualification Sampling Period Control (GPIO0 to 31)

**Figure 8-4. GPACTRL Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| QUALPRD3 | | | | | | | | QUALPRD2 | | | | | | | | QUALPRD1 | | | | | | | | QUALPRD0 | | | | | | | |
| R/W-0h | | | | | | | | R/W-0h | | | | | | | | R/W-0h | | | | | | | | R/W-0h | | | | | | | |

**Table 8-14. GPACTRL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-24 | QUALPRD3 | R/W | 0h | Qualification sampling period for GPIO24 to GPIO31:<br>0x00,QUALPRDx = PLLSYSCLK<br>0x01,QUALPRDx = PLLSYSCLK/2<br>0x02,QUALPRDx = PLLSYSCLK/4<br>....<br>0xFF,QUALPRDx = PLLSYSCLK/510<br>Reset type: SYSRSn |
| 23-16 | QUALPRD2 | R/W | 0h | Qualification sampling period for GPIO16 to GPIO23:<br>0x00,QUALPRDx = PLLSYSCLK<br>0x01,QUALPRDx = PLLSYSCLK/2<br>0x02,QUALPRDx = PLLSYSCLK/4<br>....<br>0xFF,QUALPRDx = PLLSYSCLK/510<br>Reset type: SYSRSn |
| 15-8 | QUALPRD1 | R/W | 0h | Qualification sampling period for GPIO8 to GPIO15:<br>0x00,QUALPRDx = PLLSYSCLK<br>0x01,QUALPRDx = PLLSYSCLK/2<br>0x02,QUALPRDx = PLLSYSCLK/4<br>....<br>0xFF,QUALPRDx = PLLSYSCLK/510<br>Reset type: SYSRSn |
| 7-0 | QUALPRD0 | R/W | 0h | Qualification sampling period for GPIO0 to GPIO7:<br>0x00,QUALPRDx = PLLSYSCLK<br>0x01,QUALPRDx = PLLSYSCLK/2<br>0x02,QUALPRDx = PLLSYSCLK/4<br>....<br>0xFF,QUALPRDx = PLLSYSCLK/510<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.2 GPAQSEL1 Register (Offset = 2h) [Reset = 00000000h]

GPAQSEL1 is shown in Figure 8-5 and described in Table 8-15.

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO15 | | GPIO14 | | GPIO13 | | GPIO12 | | GPIO11 | | GPIO10 | | GPIO9 | | GPIO8 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO7 | | GPIO6 | | GPIO5 | | GPIO4 | | GPIO3 | | GPIO2 | | GPIO1 | | GPIO0 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-15. GPAQSEL1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GPIO15 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 29-28 | GPIO14 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 27-26 | GPIO13 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 25-24 | GPIO12 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 23-22 | GPIO11 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 21-20 | GPIO10 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 19-18 | GPIO9 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 17-16 | GPIO8 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 15-14 | GPIO7 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 13-12 | GPIO6 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 11-10 | GPIO5 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 9-8 | GPIO4 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 7-6 | GPIO3 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 5-4 | GPIO2 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 3-2 | GPIO1 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-15. GPAQSEL1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1-0 | GPIO0 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.3 GPAQSEL2 Register (Offset = 4h) [Reset = 00000000h]

GPAQSEL2 is shown in Figure 8-6 and described in Table 8-16.

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO31 | | GPIO30 | | GPIO29 | | GPIO28 | | GPIO27 | | GPIO26 | | GPIO25 | | GPIO24 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO23 | | GPIO22 | | GPIO21 | | GPIO20 | | GPIO19 | | GPIO18 | | GPIO17 | | GPIO16 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-16. GPAQSEL2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GPIO31 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 29-28 | GPIO30 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 27-26 | GPIO29 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 25-24 | GPIO28 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 23-22 | GPIO27 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 21-20 | GPIO26 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 19-18 | GPIO25 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 17-16 | GPIO24 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 15-14 | GPIO23 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 13-12 | GPIO22 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 11-10 | GPIO21 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 9-8 | GPIO20 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 7-6 | GPIO19 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 5-4 | GPIO18 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 3-2 | GPIO17 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-16. GPAQSEL2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1-0 | GPIO16 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.4 GPAMUX1 Register (Offset = 6h) [Reset = 00000000h]

GPAMUX1 is shown in Figure 8-7 and described in Table 8-17.

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.

**Figure 8-7. GPAMUX1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO15 | | GPIO14 | | GPIO13 | | GPIO12 | | GPIO11 | | GPIO10 | | GPIO9 | | GPIO8 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO7 | | GPIO6 | | GPIO5 | | GPIO4 | | GPIO3 | | GPIO2 | | GPIO1 | | GPIO0 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-17. GPAMUX1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GPIO15 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 29-28 | GPIO14 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 27-26 | GPIO13 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 25-24 | GPIO12 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 23-22 | GPIO11 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 21-20 | GPIO10 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 19-18 | GPIO9 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 17-16 | GPIO8 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 15-14 | GPIO7 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 13-12 | GPIO6 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 11-10 | GPIO5 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 9-8 | GPIO4 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 7-6 | GPIO3 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 5-4 | GPIO2 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 3-2 | GPIO1 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-17. GPAMUX1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1-0 | GPIO0 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.5 GPAMUX2 Register (Offset = 8h) [Reset = 00000000h]

GPAMUX2 is shown in Figure 8-8 and described in Table 8-18.

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.

**Figure 8-8. GPAMUX2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO31 | | GPIO30 | | GPIO29 | | GPIO28 | | GPIO27 | | GPIO26 | | GPIO25 | | GPIO24 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO23 | | GPIO22 | | GPIO21 | | GPIO20 | | GPIO19 | | GPIO18 | | GPIO17 | | GPIO16 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-18. GPAMUX2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GPIO31 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 29-28 | GPIO30 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 27-26 | GPIO29 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 25-24 | GPIO28 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 23-22 | GPIO27 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 21-20 | GPIO26 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 19-18 | GPIO25 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 17-16 | GPIO24 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 15-14 | GPIO23 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 13-12 | GPIO22 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 11-10 | GPIO21 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 9-8 | GPIO20 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 7-6 | GPIO19 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 5-4 | GPIO18 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 3-2 | GPIO17 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-18. GPAMUX2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1-0 | GPIO16 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.6 GPADIR Register (Offset = Ah) [Reset = 00000000h]

GPADIR is shown in Figure 8-9 and described in Table 8-19.

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

**Table 8-19. GPADIR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO31 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 30 | GPIO30 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 29 | GPIO29 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 28 | GPIO28 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 27 | GPIO27 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 26 | GPIO26 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 25 | GPIO25 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 24 | GPIO24 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 23 | GPIO23 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 22 | GPIO22 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 21 | GPIO21 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-19. GPADIR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 20 | GPIO20 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 19 | GPIO19 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 18 | GPIO18 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 17 | GPIO17 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 16 | GPIO16 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 15 | GPIO15 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 14 | GPIO14 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 13 | GPIO13 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 12 | GPIO12 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 11 | GPIO11 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 10 | GPIO10 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 9 | GPIO9 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 8 | GPIO8 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 7 | GPIO7 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 6 | GPIO6 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 5 | GPIO5 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 4 | GPIO4 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 3 | GPIO3 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 2 | GPIO2 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 1 | GPIO1 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 0 | GPIO0 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.7 GPAPUD Register (Offset = Ch) [Reset = FFFFFFFFh]

GPAPUD is shown in Figure 8-10 and described in Table 8-20.

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.

**Figure 8-10. GPAPUD Register**

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

**Table 8-20. GPAPUD Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO31 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 30 | GPIO30 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 29 | GPIO29 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 28 | GPIO28 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 27 | GPIO27 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 26 | GPIO26 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 25 | GPIO25 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 24 | GPIO24 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 23 | GPIO23 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 22 | GPIO22 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 21 | GPIO21 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-20. GPAPUD Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 20 | GPIO20 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 19 | GPIO19 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 18 | GPIO18 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 17 | GPIO17 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 16 | GPIO16 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 15 | GPIO15 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 14 | GPIO14 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 13 | GPIO13 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 12 | GPIO12 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 11 | GPIO11 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 10 | GPIO10 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 9 | GPIO9 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 8 | GPIO8 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 7 | GPIO7 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 6 | GPIO6 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 5 | GPIO5 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 4 | GPIO4 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 3 | GPIO3 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 2 | GPIO2 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 1 | GPIO1 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 0 | GPIO0 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.8 GPAINV Register (Offset = 10h) [Reset = 00000000h]

GPAINV is shown in Figure 8-11 and described in Table 8-21.

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

**Table 8-21. GPAINV Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO31 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 30 | GPIO30 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 29 | GPIO29 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 28 | GPIO28 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 27 | GPIO27 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 26 | GPIO26 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 25 | GPIO25 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 24 | GPIO24 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 23 | GPIO23 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 22 | GPIO22 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 21 | GPIO21 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-21. GPAINV Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 20 | GPIO20 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 19 | GPIO19 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 18 | GPIO18 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 17 | GPIO17 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 16 | GPIO16 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 15 | GPIO15 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 14 | GPIO14 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 13 | GPIO13 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 12 | GPIO12 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 11 | GPIO11 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 10 | GPIO10 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 9 | GPIO9 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 8 | GPIO8 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 7 | GPIO7 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 6 | GPIO6 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 5 | GPIO5 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 4 | GPIO4 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 3 | GPIO3 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 2 | GPIO2 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 1 | GPIO1 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 0 | GPIO0 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.9 GPAODR Register (Offset = 12h) [Reset = 00000000h]

GPAODR is shown in Figure 8-12 and described in Table 8-22.

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

**Table 8-22. GPAODR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO31 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 30 | GPIO30 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 29 | GPIO29 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 28 | GPIO28 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 27 | GPIO27 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 26 | GPIO26 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 25 | GPIO25 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 24 | GPIO24 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 23 | GPIO23 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 22 | GPIO22 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-22. GPAODR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 21 | GPIO21 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 20 | GPIO20 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 19 | GPIO19 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 18 | GPIO18 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 17 | GPIO17 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 16 | GPIO16 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 15 | GPIO15 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 14 | GPIO14 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 13 | GPIO13 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 12 | GPIO12 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 11 | GPIO11 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 10 | GPIO10 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 9 | GPIO9 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 8 | GPIO8 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 7 | GPIO7 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 6 | GPIO6 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 5 | GPIO5 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 4 | GPIO4 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 3 | GPIO3 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 2 | GPIO2 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 1 | GPIO1 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 0 | GPIO0 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.10 GPAGMUX1 Register (Offset = 20h) [Reset = 00000000h]

GPAGMUX1 is shown in Figure 8-13 and described in Table 8-23.

Return to the Summary Table.

GPIO A Peripheral Group Mux (GPIO0 to 15)
Defines pin-muxing selection for GPIO.
Note: For complete pin-mux selection on GPIOx, GPAMUXy.GPIOx configuration is also required.

**Figure 8-13. GPAGMUX1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO15 | | GPIO14 | | GPIO13 | | GPIO12 | | GPIO11 | | GPIO10 | | GPIO9 | | GPIO8 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO7 | | GPIO6 | | GPIO5 | | GPIO4 | | GPIO3 | | GPIO2 | | GPIO1 | | GPIO0 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-23. GPAGMUX1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GPIO15 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 29-28 | GPIO14 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 27-26 | GPIO13 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 25-24 | GPIO12 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 23-22 | GPIO11 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 21-20 | GPIO10 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 19-18 | GPIO9 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 17-16 | GPIO8 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 15-14 | GPIO7 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 13-12 | GPIO6 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 11-10 | GPIO5 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 9-8 | GPIO4 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 7-6 | GPIO3 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 5-4 | GPIO2 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 3-2 | GPIO1 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 1-0 | GPIO0 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.11 GPAGMUX2 Register (Offset = 22h) [Reset = 00000000h]

GPAGMUX2 is shown in Figure 8-14 and described in Table 8-24.

Return to the Summary Table.

GPIO A Peripheral Group Mux (GPIO16 to 31)
Defines pin-muxing selection for GPIO.
Note: For complete pin-mux selection on GPIOx, GPAMUXy.GPIOx configuration is also required.

**Figure 8-14. GPAGMUX2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO31 | | GPIO30 | | GPIO29 | | GPIO28 | | GPIO27 | | GPIO26 | | GPIO25 | | GPIO24 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO23 | | GPIO22 | | GPIO21 | | GPIO20 | | GPIO19 | | GPIO18 | | GPIO17 | | GPIO16 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-24. GPAGMUX2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GPIO31 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 29-28 | GPIO30 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 27-26 | GPIO29 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 25-24 | GPIO28 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 23-22 | GPIO27 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 21-20 | GPIO26 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 19-18 | GPIO25 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 17-16 | GPIO24 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 15-14 | GPIO23 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 13-12 | GPIO22 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 11-10 | GPIO21 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 9-8 | GPIO20 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 7-6 | GPIO19 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 5-4 | GPIO18 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 3-2 | GPIO17 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 1-0 | GPIO16 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.12 GPACSEL1 Register (Offset = 28h) [Reset = 00000000h]

GPACSEL1 is shown in Figure 8-15 and described in Table 8-25.

Return to the Summary Table.

GPIO A Core Select Register (GPIO0 to 7)
Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
xx00: CPU1 selected
xx01: CPU1.CLA1 selected
xx10: CPU2 selected
xx11: CPU2.CLA1 selected

**Figure 8-15. GPACSEL1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO7 | | | | GPIO6 | | | | GPIO5 | | | | GPIO4 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO3 | | | | GPIO2 | | | | GPIO1 | | | | GPIO0 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |

**Table 8-25. GPACSEL1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-28 | GPIO7 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 27-24 | GPIO6 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 23-20 | GPIO5 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 19-16 | GPIO4 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 15-12 | GPIO3 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 11-8 | GPIO2 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 7-4 | GPIO1 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 3-0 | GPIO0 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.13 GPACSEL2 Register (Offset = 2Ah) [Reset = 00000000h]

GPACSEL2 is shown in Figure 8-16 and described in Table 8-26.

Return to the Summary Table.

GPIO A Core Select Register (GPIO8 to 15)
Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
xx00: CPU1 selected
xx01: CPU1.CLA1 selected
xx10: CPU2 selected
xx11: CPU2.CLA1 selected

**Figure 8-16. GPACSEL2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO15 | | | | GPIO14 | | | | GPIO13 | | | | GPIO12 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO11 | | | | GPIO10 | | | | GPIO9 | | | | GPIO8 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |

**Table 8-26. GPACSEL2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-28 | GPIO15 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 27-24 | GPIO14 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 23-20 | GPIO13 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 19-16 | GPIO12 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 15-12 | GPIO11 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 11-8 | GPIO10 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 7-4 | GPIO9 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 3-0 | GPIO8 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.14 GPACSEL3 Register (Offset = 2Ch) [Reset = 00000000h]

GPACSEL3 is shown in Figure 8-17 and described in Table 8-27.

Return to the Summary Table.

GPIO A Core Select Register (GPIO16 to 23)
Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
xx00: CPU1 selected
xx01: CPU1.CLA1 selected
xx10: CPU2 selected
xx11: CPU2.CLA1 selected

**Figure 8-17. GPACSEL3 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO23 | | | | GPIO22 | | | | GPIO21 | | | | GPIO20 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO19 | | | | GPIO18 | | | | GPIO17 | | | | GPIO16 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |

**Table 8-27. GPACSEL3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-28 | GPIO23 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 27-24 | GPIO22 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 23-20 | GPIO21 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 19-16 | GPIO20 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 15-12 | GPIO19 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 11-8 | GPIO18 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 7-4 | GPIO17 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 3-0 | GPIO16 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.15 GPACSEL4 Register (Offset = 2Eh) [Reset = 00000000h]

GPACSEL4 is shown in Figure 8-18 and described in Table 8-28.

Return to the Summary Table.

GPIO A Core Select Register (GPIO24 to 31)
Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
xx00: CPU1 selected
xx01: CPU1.CLA1 selected
xx10: CPU2 selected
xx11: CPU2.CLA1 selected

**Figure 8-18. GPACSEL4 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO31 | | | | GPIO30 | | | | GPIO29 | | | | GPIO28 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO27 | | | | GPIO26 | | | | GPIO25 | | | | GPIO24 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |

**Table 8-28. GPACSEL4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-28 | GPIO31 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 27-24 | GPIO30 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 23-20 | GPIO29 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 19-16 | GPIO28 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 15-12 | GPIO27 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 11-8 | GPIO26 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 7-4 | GPIO25 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 3-0 | GPIO24 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.16 GPALOCK Register (Offset = 3Ch) [Reset = 00000000h]

GPALOCK is shown in Figure 8-19 and described in Table 8-29.

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 8-19. GPALOCK Register**

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

**Table 8-29. GPALOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO31 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 30 | GPIO30 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 29 | GPIO29 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 28 | GPIO28 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 27 | GPIO27 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 26 | GPIO26 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 25 | GPIO25 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 24 | GPIO24 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 23 | GPIO23 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 22 | GPIO22 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 21 | GPIO21 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-29. GPALOCK Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 20 | GPIO20 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 19 | GPIO19 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 18 | GPIO18 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 17 | GPIO17 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 16 | GPIO16 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 15 | GPIO15 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 14 | GPIO14 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 13 | GPIO13 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 12 | GPIO12 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 11 | GPIO11 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 10 | GPIO10 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 9 | GPIO9 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 8 | GPIO8 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 7 | GPIO7 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 6 | GPIO6 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 5 | GPIO5 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 4 | GPIO4 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 3 | GPIO3 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 2 | GPIO2 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 1 | GPIO1 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 0 | GPIO0 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.17 GPACR Register (Offset = 3Eh) [Reset = 00000000h]

GPACR is shown in Figure 8-20 and described in Table 8-30.

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 8-20. GPACR Register**

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

**Table 8-30. GPACR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO31 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 30 | GPIO30 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 29 | GPIO29 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 28 | GPIO28 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 27 | GPIO27 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 26 | GPIO26 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 25 | GPIO25 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 24 | GPIO24 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 23 | GPIO23 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 22 | GPIO22 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 21 | GPIO21 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 20 | GPIO20 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-30. GPACR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19 | GPIO19 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 18 | GPIO18 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 17 | GPIO17 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 16 | GPIO16 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 15 | GPIO15 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 14 | GPIO14 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 13 | GPIO13 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 12 | GPIO12 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 11 | GPIO11 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 10 | GPIO10 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 9 | GPIO9 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 8 | GPIO8 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 7 | GPIO7 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 6 | GPIO6 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 5 | GPIO5 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 4 | GPIO4 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 3 | GPIO3 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 2 | GPIO2 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 1 | GPIO1 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 0 | GPIO0 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.18 GPBCTRL Register (Offset = 40h) [Reset = 00000000h]

GPBCTRL is shown in Figure 8-21 and described in Table 8-31.

Return to the Summary Table.

GPIO B Qualification Sampling Period Control (GPIO32 to 63)

**Figure 8-21. GPBCTRL Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| QUALPRD3 | | | | | | | | QUALPRD2 | | | | | | | | QUALPRD1 | | | | | | | | QUALPRD0 | | | | | | | |
| R/W-0h | | | | | | | | R/W-0h | | | | | | | | R/W-0h | | | | | | | | R/W-0h | | | | | | | |

**Table 8-31. GPBCTRL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-24 | QUALPRD3 | R/W | 0h | Qualification sampling period for GPIO56 to GPIO63:<br>0x00,QUALPRDx = PLLSYSCLK<br>0x01,QUALPRDx = PLLSYSCLK/2<br>0x02,QUALPRDx = PLLSYSCLK/4<br>....<br>0xFF,QUALPRDx = PLLSYSCLK/510<br>Reset type: SYSRSn |
| 23-16 | QUALPRD2 | R/W | 0h | Qualification sampling period for GPIO48 to GPIO55:<br>0x00,QUALPRDx = PLLSYSCLK<br>0x01,QUALPRDx = PLLSYSCLK/2<br>0x02,QUALPRDx = PLLSYSCLK/4<br>....<br>0xFF,QUALPRDx = PLLSYSCLK/510<br>Reset type: SYSRSn |
| 15-8 | QUALPRD1 | R/W | 0h | Qualification sampling period for GPIO40 to GPIO47:<br>0x00,QUALPRDx = PLLSYSCLK<br>0x01,QUALPRDx = PLLSYSCLK/2<br>0x02,QUALPRDx = PLLSYSCLK/4<br>....<br>0xFF,QUALPRDx = PLLSYSCLK/510<br>Reset type: SYSRSn |
| 7-0 | QUALPRD0 | R/W | 0h | Qualification sampling period for GPIO32 to GPIO39:<br>0x00,QUALPRDx = PLLSYSCLK<br>0x01,QUALPRDx = PLLSYSCLK/2<br>0x02,QUALPRDx = PLLSYSCLK/4<br>....<br>0xFF,QUALPRDx = PLLSYSCLK/510<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.19 GPBQSEL1 Register (Offset = 42h) [Reset = 00000000h]

GPBQSEL1 is shown in Figure 8-22 and described in Table 8-32.

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 8-22. GPBQSEL1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO47 | | GPIO46 | | GPIO45 | | GPIO44 | | GPIO43 | | GPIO42 | | GPIO41 | | GPIO40 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO39 | | GPIO38 | | GPIO37 | | GPIO36 | | GPIO35 | | GPIO34 | | GPIO33 | | GPIO32 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-32. GPBQSEL1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GPIO47 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 29-28 | GPIO46 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 27-26 | GPIO45 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 25-24 | GPIO44 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 23-22 | GPIO43 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 21-20 | GPIO42 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 19-18 | GPIO41 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 17-16 | GPIO40 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 15-14 | GPIO39 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 13-12 | GPIO38 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 11-10 | GPIO37 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 9-8 | GPIO36 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 7-6 | GPIO35 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 5-4 | GPIO34 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 3-2 | GPIO33 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-32. GPBQSEL1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1-0 | GPIO32 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.20 GPBQSEL2 Register (Offset = 44h) [Reset = 00000000h]

GPBQSEL2 is shown in Figure 8-23 and described in Table 8-33.

Return to the Summary Table.

GPIO B Qualifier Select 2 Register (GPIO48 to 63)
Input qualification type:
0,0 Sync
0,1 Qualification (3 samples)
1,0 Qualification (6 samples)
1,1 Async (no Sync or Qualification)

**Figure 8-23. GPBQSEL2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO63 | | GPIO62 | | GPIO61 | | GPIO60 | | GPIO59 | | GPIO58 | | GPIO57 | | GPIO56 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO55 | | GPIO54 | | GPIO53 | | GPIO52 | | GPIO51 | | GPIO50 | | GPIO49 | | GPIO48 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-33. GPBQSEL2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GPIO63 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 29-28 | GPIO62 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 27-26 | GPIO61 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 25-24 | GPIO60 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 23-22 | GPIO59 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 21-20 | GPIO58 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 19-18 | GPIO57 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 17-16 | GPIO56 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 15-14 | GPIO55 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 13-12 | GPIO54 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 11-10 | GPIO53 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 9-8 | GPIO52 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 7-6 | GPIO51 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 5-4 | GPIO50 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 3-2 | GPIO49 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-33. GPBQSEL2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1-0 | GPIO48 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.21 GPBMUX1 Register (Offset = 46h) [Reset = 00000000h]

GPBMUX1 is shown in Figure 8-24 and described in Table 8-34.

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.

**Figure 8-24. GPBMUX1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO47 | | GPIO46 | | GPIO45 | | GPIO44 | | GPIO43 | | GPIO42 | | GPIO41 | | GPIO40 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO39 | | GPIO38 | | GPIO37 | | GPIO36 | | GPIO35 | | GPIO34 | | GPIO33 | | GPIO32 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-34. GPBMUX1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GPIO47 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 29-28 | GPIO46 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 27-26 | GPIO45 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 25-24 | GPIO44 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 23-22 | GPIO43 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 21-20 | GPIO42 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 19-18 | GPIO41 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 17-16 | GPIO40 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 15-14 | GPIO39 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 13-12 | GPIO38 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 11-10 | GPIO37 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 9-8 | GPIO36 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 7-6 | GPIO35 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 5-4 | GPIO34 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 3-2 | GPIO33 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-34. GPBMUX1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1-0 | GPIO32 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.22 GPBMUX2 Register (Offset = 48h) [Reset = 00000000h]

GPBMUX2 is shown in Figure 8-25 and described in Table 8-35.

Return to the Summary Table.

GPIO B Mux 2 Register (GPIO48 to 63)
Defines pin-muxing selection for GPIO.
Notes:
The respective GPyGMUXn.GPIOz must be configured prior to this register to avoid intermediate peripheral selects being mapped to the GPIO.

**Figure 8-25. GPBMUX2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO63 | | GPIO62 | | GPIO61 | | GPIO60 | | GPIO59 | | GPIO58 | | GPIO57 | | GPIO56 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO55 | | GPIO54 | | GPIO53 | | GPIO52 | | GPIO51 | | GPIO50 | | GPIO49 | | GPIO48 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-35. GPBMUX2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GPIO63 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 29-28 | GPIO62 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 27-26 | GPIO61 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 25-24 | GPIO60 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 23-22 | GPIO59 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 21-20 | GPIO58 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 19-18 | GPIO57 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 17-16 | GPIO56 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 15-14 | GPIO55 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 13-12 | GPIO54 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 11-10 | GPIO53 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 9-8 | GPIO52 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 7-6 | GPIO51 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 5-4 | GPIO50 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 3-2 | GPIO49 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-35. GPBMUX2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1-0 | GPIO48 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.23 GPBDIR Register (Offset = 4Ah) [Reset = 00000000h]

GPBDIR is shown in Figure 8-26 and described in Table 8-36.

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 8-26. GPBDIR Register**

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

**Table 8-36. GPBDIR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO63 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 30 | GPIO62 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 29 | GPIO61 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 28 | GPIO60 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 27 | GPIO59 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 26 | GPIO58 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 25 | GPIO57 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 24 | GPIO56 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 23 | GPIO55 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 22 | GPIO54 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 21 | GPIO53 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-36. GPBDIR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 20 | GPIO52 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 19 | GPIO51 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 18 | GPIO50 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 17 | GPIO49 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 16 | GPIO48 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 15 | GPIO47 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 14 | GPIO46 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 13 | GPIO45 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 12 | GPIO44 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 11 | GPIO43 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 10 | GPIO42 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 9 | GPIO41 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 8 | GPIO40 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 7 | GPIO39 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 6 | GPIO38 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 5 | GPIO37 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 4 | GPIO36 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 3 | GPIO35 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 2 | GPIO34 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 1 | GPIO33 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 0 | GPIO32 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.24 GPBPUD Register (Offset = 4Ch) [Reset = FFFFFFFFh]

GPBPUD is shown in Figure 8-27 and described in Table 8-37.

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.

**Figure 8-27. GPBPUD Register**

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

**Table 8-37. GPBPUD Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO63 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 30 | GPIO62 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 29 | GPIO61 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 28 | GPIO60 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 27 | GPIO59 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 26 | GPIO58 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 25 | GPIO57 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 24 | GPIO56 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 23 | GPIO55 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 22 | GPIO54 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 21 | GPIO53 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-37. GPBPUD Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 20 | GPIO52 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 19 | GPIO51 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 18 | GPIO50 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 17 | GPIO49 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 16 | GPIO48 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 15 | GPIO47 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 14 | GPIO46 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 13 | GPIO45 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 12 | GPIO44 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 11 | GPIO43 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 10 | GPIO42 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 9 | GPIO41 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 8 | GPIO40 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 7 | GPIO39 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 6 | GPIO38 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 5 | GPIO37 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 4 | GPIO36 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 3 | GPIO35 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 2 | GPIO34 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 1 | GPIO33 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 0 | GPIO32 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.25 GPBINV Register (Offset = 50h) [Reset = 00000000h]

GPBINV is shown in Figure 8-28 and described in Table 8-38.

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 8-28. GPBINV Register**

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

**Table 8-38. GPBINV Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO63 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 30 | GPIO62 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 29 | GPIO61 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 28 | GPIO60 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 27 | GPIO59 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 26 | GPIO58 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 25 | GPIO57 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 24 | GPIO56 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 23 | GPIO55 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 22 | GPIO54 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 21 | GPIO53 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-38. GPBINV Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 20 | GPIO52 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 19 | GPIO51 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 18 | GPIO50 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 17 | GPIO49 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 16 | GPIO48 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 15 | GPIO47 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 14 | GPIO46 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 13 | GPIO45 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 12 | GPIO44 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 11 | GPIO43 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 10 | GPIO42 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 9 | GPIO41 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 8 | GPIO40 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 7 | GPIO39 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 6 | GPIO38 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 5 | GPIO37 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 4 | GPIO36 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 3 | GPIO35 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 2 | GPIO34 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 1 | GPIO33 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 0 | GPIO32 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.26 GPBODR Register (Offset = 52h) [Reset = 00000000h]

GPBODR is shown in Figure 8-29 and described in Table 8-39.

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 8-29. GPBODR Register**

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

**Table 8-39. GPBODR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO63 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 30 | GPIO62 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 29 | GPIO61 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 28 | GPIO60 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 27 | GPIO59 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 26 | GPIO58 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 25 | GPIO57 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 24 | GPIO56 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 23 | GPIO55 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 22 | GPIO54 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-39. GPBODR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 21 | GPIO53 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 20 | GPIO52 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 19 | GPIO51 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 18 | GPIO50 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 17 | GPIO49 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 16 | GPIO48 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 15 | GPIO47 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 14 | GPIO46 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 13 | GPIO45 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 12 | GPIO44 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 11 | GPIO43 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 10 | GPIO42 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 9 | GPIO41 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 8 | GPIO40 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 7 | GPIO39 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 6 | GPIO38 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 5 | GPIO37 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 4 | GPIO36 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 3 | GPIO35 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 2 | GPIO34 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 1 | GPIO33 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 0 | GPIO32 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.27 GPBAMSEL Register (Offset = 54h) [Reset = 00000000h]

GPBAMSEL is shown in Figure 8-30 and described in Table 8-40.

Return to the Summary Table.

GPIO B Analog Mode Select register
Selects between digital and analog functionality for GPIO pins.
0: The pin is configured to digital functions according to the other GPIO configuration registers
1: The analog function of the pin is enabled

**Figure 8-30. GPBAMSEL Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | 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 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | 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 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | RESERVED | RESERVED | RESERVED | GPIO43 | GPIO42 | 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 | RESERVED | RESERVED | 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 |

**Table 8-40. GPBAMSEL 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 | GPIO43 | R/W | 0h | Selects the USB0DP function<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-40. GPBAMSEL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 10 | GPIO42 | R/W | 0h | Selects the USB0DM function<br>Reset type: SYSRSn |
| 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 |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.28 GPBGMUX1 Register (Offset = 60h) [Reset = 00000000h]

GPBGMUX1 is shown in Figure 8-31 and described in Table 8-41.

Return to the Summary Table.

GPIO B Peripheral Group Mux (GPIO32 to 47)
Defines pin-muxing selection for GPIO.
Note: For complete pin-mux selection on GPIOx, GPAMUXy.GPIOx configuration is also required.

**Figure 8-31. GPBGMUX1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO47 | | GPIO46 | | GPIO45 | | GPIO44 | | GPIO43 | | GPIO42 | | GPIO41 | | GPIO40 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO39 | | GPIO38 | | GPIO37 | | GPIO36 | | GPIO35 | | GPIO34 | | GPIO33 | | GPIO32 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-41. GPBGMUX1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GPIO47 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 29-28 | GPIO46 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 27-26 | GPIO45 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 25-24 | GPIO44 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 23-22 | GPIO43 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 21-20 | GPIO42 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 19-18 | GPIO41 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 17-16 | GPIO40 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 15-14 | GPIO39 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 13-12 | GPIO38 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 11-10 | GPIO37 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 9-8 | GPIO36 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 7-6 | GPIO35 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 5-4 | GPIO34 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 3-2 | GPIO33 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 1-0 | GPIO32 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.29 GPBGMUX2 Register (Offset = 62h) [Reset = 00000000h]

GPBGMUX2 is shown in Figure 8-32 and described in Table 8-42.

Return to the Summary Table.

GPIO B Peripheral Group Mux (GPIO48 to 63)
Defines pin-muxing selection for GPIO.
Note: For complete pin-mux selection on GPIOx, GPAMUXy.GPIOx configuration is also required.

**Figure 8-32. GPBGMUX2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO63 | | GPIO62 | | GPIO61 | | GPIO60 | | GPIO59 | | GPIO58 | | GPIO57 | | GPIO56 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO55 | | GPIO54 | | GPIO53 | | GPIO52 | | GPIO51 | | GPIO50 | | GPIO49 | | GPIO48 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-42. GPBGMUX2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GPIO63 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 29-28 | GPIO62 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 27-26 | GPIO61 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 25-24 | GPIO60 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 23-22 | GPIO59 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 21-20 | GPIO58 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 19-18 | GPIO57 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 17-16 | GPIO56 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 15-14 | GPIO55 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 13-12 | GPIO54 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 11-10 | GPIO53 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 9-8 | GPIO52 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 7-6 | GPIO51 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 5-4 | GPIO50 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 3-2 | GPIO49 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 1-0 | GPIO48 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.30 GPBCSEL1 Register (Offset = 68h) [Reset = 00000000h]

GPBCSEL1 is shown in Figure 8-33 and described in Table 8-43.

Return to the Summary Table.

GPIO B Core Select Register (GPIO32 to 39)
Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
xx00: CPU1 selected
xx01: CPU1.CLA1 selected
xx10: CPU2 selected
xx11: CPU2.CLA1 selected

**Figure 8-33. GPBCSEL1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO39 | | | | GPIO38 | | | | GPIO37 | | | | GPIO36 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO35 | | | | GPIO34 | | | | GPIO33 | | | | GPIO32 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |

**Table 8-43. GPBCSEL1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-28 | GPIO39 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 27-24 | GPIO38 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 23-20 | GPIO37 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 19-16 | GPIO36 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 15-12 | GPIO35 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 11-8 | GPIO34 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 7-4 | GPIO33 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 3-0 | GPIO32 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.31 GPBCSEL2 Register (Offset = 6Ah) [Reset = 00000000h]

GPBCSEL2 is shown in Figure 8-34 and described in Table 8-44.

Return to the Summary Table.

GPIO B Core Select Register (GPIO40 to 47)
Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
xx00: CPU1 selected
xx01: CPU1.CLA1 selected
xx10: CPU2 selected
xx11: CPU2.CLA1 selected

**Figure 8-34. GPBCSEL2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO47 | | | | GPIO46 | | | | GPIO45 | | | | GPIO44 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO43 | | | | GPIO42 | | | | GPIO41 | | | | GPIO40 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |

**Table 8-44. GPBCSEL2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-28 | GPIO47 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 27-24 | GPIO46 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 23-20 | GPIO45 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 19-16 | GPIO44 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 15-12 | GPIO43 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 11-8 | GPIO42 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 7-4 | GPIO41 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 3-0 | GPIO40 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.32 GPBCSEL3 Register (Offset = 6Ch) [Reset = 00000000h]

GPBCSEL3 is shown in Figure 8-35 and described in Table 8-45.

Return to the Summary Table.

GPIO B Core Select Register (GPIO48 to 55)
Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
xx00: CPU1 selected
xx01: CPU1.CLA1 selected
xx10: CPU2 selected
xx11: CPU2.CLA1 selected

**Figure 8-35. GPBCSEL3 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO55 | | | | GPIO54 | | | | GPIO53 | | | | GPIO52 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO51 | | | | GPIO50 | | | | GPIO49 | | | | GPIO48 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |

**Table 8-45. GPBCSEL3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-28 | GPIO55 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 27-24 | GPIO54 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 23-20 | GPIO53 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 19-16 | GPIO52 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 15-12 | GPIO51 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 11-8 | GPIO50 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 7-4 | GPIO49 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 3-0 | GPIO48 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.33 GPBCSEL4 Register (Offset = 6Eh) [Reset = 00000000h]

GPBCSEL4 is shown in Figure 8-36 and described in Table 8-46.

Return to the Summary Table.

GPIO B Core Select Register (GPIO56 to 63)
Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
xx00: CPU1 selected
xx01: CPU1.CLA1 selected
xx10: CPU2 selected
xx11: CPU2.CLA1 selected

**Figure 8-36. GPBCSEL4 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO63 | | | | GPIO62 | | | | GPIO61 | | | | GPIO60 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO59 | | | | GPIO58 | | | | GPIO57 | | | | GPIO56 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |

**Table 8-46. GPBCSEL4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-28 | GPIO63 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 27-24 | GPIO62 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 23-20 | GPIO61 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 19-16 | GPIO60 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 15-12 | GPIO59 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 11-8 | GPIO58 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 7-4 | GPIO57 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 3-0 | GPIO56 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.34 GPBLOCK Register (Offset = 7Ch) [Reset = 00000000h]

GPBLOCK is shown in Figure 8-37 and described in Table 8-47.

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 8-37. GPBLOCK Register**

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

**Table 8-47. GPBLOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO63 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 30 | GPIO62 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 29 | GPIO61 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 28 | GPIO60 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 27 | GPIO59 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 26 | GPIO58 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 25 | GPIO57 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 24 | GPIO56 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 23 | GPIO55 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 22 | GPIO54 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 21 | GPIO53 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-47. GPBLOCK Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 20 | GPIO52 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 19 | GPIO51 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 18 | GPIO50 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 17 | GPIO49 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 16 | GPIO48 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 15 | GPIO47 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 14 | GPIO46 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 13 | GPIO45 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 12 | GPIO44 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 11 | GPIO43 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 10 | GPIO42 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 9 | GPIO41 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 8 | GPIO40 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 7 | GPIO39 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 6 | GPIO38 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 5 | GPIO37 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 4 | GPIO36 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 3 | GPIO35 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 2 | GPIO34 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 1 | GPIO33 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 0 | GPIO32 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.35 GPBCR Register (Offset = 7Eh) [Reset = 00000000h]

GPBCR is shown in Figure 8-38 and described in Table 8-48.

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 8-38. GPBCR Register**

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

**Table 8-48. GPBCR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO63 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 30 | GPIO62 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 29 | GPIO61 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 28 | GPIO60 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 27 | GPIO59 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 26 | GPIO58 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 25 | GPIO57 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 24 | GPIO56 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 23 | GPIO55 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 22 | GPIO54 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 21 | GPIO53 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 20 | GPIO52 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-48. GPBCR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19 | GPIO51 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 18 | GPIO50 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 17 | GPIO49 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 16 | GPIO48 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 15 | GPIO47 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 14 | GPIO46 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 13 | GPIO45 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 12 | GPIO44 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 11 | GPIO43 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 10 | GPIO42 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 9 | GPIO41 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 8 | GPIO40 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 7 | GPIO39 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 6 | GPIO38 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 5 | GPIO37 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 4 | GPIO36 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 3 | GPIO35 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 2 | GPIO34 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 1 | GPIO33 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 0 | GPIO32 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.36 GPCCTRL Register (Offset = 80h) [Reset = 00000000h]

GPCCTRL is shown in Figure 8-39 and described in Table 8-49.

Return to the Summary Table.

GPIO C Qualification Sampling Period Control (GPIO64 to 95)

**Figure 8-39. GPCCTRL Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| QUALPRD3 | | | | | | | | QUALPRD2 | | | | | | | | QUALPRD1 | | | | | | | | QUALPRD0 | | | | | | | |
| R/W-0h | | | | | | | | R/W-0h | | | | | | | | R/W-0h | | | | | | | | R/W-0h | | | | | | | |

**Table 8-49. GPCCTRL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-24 | QUALPRD3 | R/W | 0h | Qualification sampling period for GPIO88 to GPIO95:<br>0x00,QUALPRDx = PLLSYSCLK<br>0x01,QUALPRDx = PLLSYSCLK/2<br>0x02,QUALPRDx = PLLSYSCLK/4<br>....<br>0xFF,QUALPRDx = PLLSYSCLK/510<br>Reset type: SYSRSn |
| 23-16 | QUALPRD2 | R/W | 0h | Qualification sampling period for GPIO80 to GPIO87:<br>0x00,QUALPRDx = PLLSYSCLK<br>0x01,QUALPRDx = PLLSYSCLK/2<br>0x02,QUALPRDx = PLLSYSCLK/4<br>....<br>0xFF,QUALPRDx = PLLSYSCLK/510<br>Reset type: SYSRSn |
| 15-8 | QUALPRD1 | R/W | 0h | Qualification sampling period for GPIO72 to GPIO79:<br>0x00,QUALPRDx = PLLSYSCLK<br>0x01,QUALPRDx = PLLSYSCLK/2<br>0x02,QUALPRDx = PLLSYSCLK/4<br>....<br>0xFF,QUALPRDx = PLLSYSCLK/510<br>Reset type: SYSRSn |
| 7-0 | QUALPRD0 | R/W | 0h | Qualification sampling period for GPIO64 to GPIO71:<br>0x00,QUALPRDx = PLLSYSCLK<br>0x01,QUALPRDx = PLLSYSCLK/2<br>0x02,QUALPRDx = PLLSYSCLK/4<br>....<br>0xFF,QUALPRDx = PLLSYSCLK/510<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.37 GPCQSEL1 Register (Offset = 82h) [Reset = 00000000h]

GPCQSEL1 is shown in Figure 8-40 and described in Table 8-50.

Return to the Summary Table.

GPIO C Qualifier Select 1 Register (GPIO64 to 79)
Input qualification type:
0,0 Sync
0,1 Qualification (3 samples)
1,0 Qualification (6 samples)
1,1 Async (no Sync or Qualification)

**Figure 8-40. GPCQSEL1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO79 | | GPIO78 | | GPIO77 | | GPIO76 | | GPIO75 | | GPIO74 | | GPIO73 | | GPIO72 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO71 | | GPIO70 | | GPIO69 | | GPIO68 | | GPIO67 | | GPIO66 | | GPIO65 | | GPIO64 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-50. GPCQSEL1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GPIO79 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 29-28 | GPIO78 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 27-26 | GPIO77 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 25-24 | GPIO76 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 23-22 | GPIO75 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 21-20 | GPIO74 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 19-18 | GPIO73 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 17-16 | GPIO72 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 15-14 | GPIO71 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 13-12 | GPIO70 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 11-10 | GPIO69 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 9-8 | GPIO68 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 7-6 | GPIO67 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 5-4 | GPIO66 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 3-2 | GPIO65 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-50. GPCQSEL1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1-0 | GPIO64 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.38 GPCQSEL2 Register (Offset = 84h) [Reset = 00000000h]

GPCQSEL2 is shown in Figure 8-41 and described in Table 8-51.

Return to the Summary Table.

GPIO C Qualifier Select 2 Register (GPIO80 to 95)
Input qualification type:
0,0 Sync
0,1 Qualification (3 samples)
1,0 Qualification (6 samples)
1,1 Async (no Sync or Qualification)

**Figure 8-41. GPCQSEL2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO95 | | GPIO94 | | GPIO93 | | GPIO92 | | GPIO91 | | GPIO90 | | GPIO89 | | GPIO88 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO87 | | GPIO86 | | GPIO85 | | GPIO84 | | GPIO83 | | GPIO82 | | GPIO81 | | GPIO80 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-51. GPCQSEL2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GPIO95 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 29-28 | GPIO94 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 27-26 | GPIO93 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 25-24 | GPIO92 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 23-22 | GPIO91 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 21-20 | GPIO90 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 19-18 | GPIO89 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 17-16 | GPIO88 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 15-14 | GPIO87 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 13-12 | GPIO86 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 11-10 | GPIO85 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 9-8 | GPIO84 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 7-6 | GPIO83 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 5-4 | GPIO82 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 3-2 | GPIO81 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-51. GPCQSEL2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1-0 | GPIO80 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.39 GPCMUX1 Register (Offset = 86h) [Reset = 00000000h]

GPCMUX1 is shown in Figure 8-42 and described in Table 8-52.

Return to the Summary Table.

GPIO C Mux 1 Register (GPIO64 to 79)
Defines pin-muxing selection for GPIO.
Notes:
The respective GPyGMUXn.GPIOz must be configured prior to this register to avoid intermediate peripheral selects being mapped to the GPIO.

**Figure 8-42. GPCMUX1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO79 | | GPIO78 | | GPIO77 | | GPIO76 | | GPIO75 | | GPIO74 | | GPIO73 | | GPIO72 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO71 | | GPIO70 | | GPIO69 | | GPIO68 | | GPIO67 | | GPIO66 | | GPIO65 | | GPIO64 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-52. GPCMUX1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GPIO79 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 29-28 | GPIO78 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 27-26 | GPIO77 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 25-24 | GPIO76 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 23-22 | GPIO75 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 21-20 | GPIO74 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 19-18 | GPIO73 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 17-16 | GPIO72 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 15-14 | GPIO71 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 13-12 | GPIO70 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 11-10 | GPIO69 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 9-8 | GPIO68 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 7-6 | GPIO67 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 5-4 | GPIO66 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 3-2 | GPIO65 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-52. GPCMUX1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1-0 | GPIO64 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.40 GPCMUX2 Register (Offset = 88h) [Reset = 00000000h]

GPCMUX2 is shown in Figure 8-43 and described in Table 8-53.

Return to the Summary Table.

GPIO C Mux 2 Register (GPIO80 to 95)
Defines pin-muxing selection for GPIO.
Notes:
The respective GPyGMUXn.GPIOz must be configured prior to this register to avoid intermediate peripheral selects being mapped to the GPIO.

**Figure 8-43. GPCMUX2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO95 | | GPIO94 | | GPIO93 | | GPIO92 | | GPIO91 | | GPIO90 | | GPIO89 | | GPIO88 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO87 | | GPIO86 | | GPIO85 | | GPIO84 | | GPIO83 | | GPIO82 | | GPIO81 | | GPIO80 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-53. GPCMUX2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GPIO95 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 29-28 | GPIO94 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 27-26 | GPIO93 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 25-24 | GPIO92 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 23-22 | GPIO91 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 21-20 | GPIO90 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 19-18 | GPIO89 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 17-16 | GPIO88 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 15-14 | GPIO87 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 13-12 | GPIO86 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 11-10 | GPIO85 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 9-8 | GPIO84 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 7-6 | GPIO83 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 5-4 | GPIO82 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 3-2 | GPIO81 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-53. GPCMUX2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1-0 | GPIO80 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.41 GPCDIR Register (Offset = 8Ah) [Reset = 00000000h]

GPCDIR is shown in Figure 8-44 and described in Table 8-54.

Return to the Summary Table.

GPIO C Direction Register (GPIO64 to 95)
Controls direction of GPIO pins when the specified pin is configured in GPIO mode.
0: Configures pin as input.
1: Configures pin as output.
Reading the register returns the current value of the register setting.

**Figure 8-44. GPCDIR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO95 | GPIO94 | GPIO93 | GPIO92 | GPIO91 | GPIO90 | GPIO89 | GPIO88 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO87 | GPIO86 | GPIO85 | GPIO84 | GPIO83 | GPIO82 | GPIO81 | GPIO80 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO79 | GPIO78 | GPIO77 | GPIO76 | GPIO75 | GPIO74 | GPIO73 | GPIO72 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO71 | GPIO70 | GPIO69 | GPIO68 | GPIO67 | GPIO66 | GPIO65 | GPIO64 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 8-54. GPCDIR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO95 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 30 | GPIO94 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 29 | GPIO93 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 28 | GPIO92 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 27 | GPIO91 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 26 | GPIO90 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 25 | GPIO89 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 24 | GPIO88 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 23 | GPIO87 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 22 | GPIO86 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 21 | GPIO85 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-54. GPCDIR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 20 | GPIO84 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 19 | GPIO83 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 18 | GPIO82 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 17 | GPIO81 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 16 | GPIO80 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 15 | GPIO79 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 14 | GPIO78 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 13 | GPIO77 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 12 | GPIO76 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 11 | GPIO75 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 10 | GPIO74 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 9 | GPIO73 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 8 | GPIO72 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 7 | GPIO71 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 6 | GPIO70 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 5 | GPIO69 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 4 | GPIO68 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 3 | GPIO67 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 2 | GPIO66 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 1 | GPIO65 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 0 | GPIO64 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.42 GPCPUD Register (Offset = 8Ch) [Reset = FFFFFFFFh]

GPCPUD is shown in Figure 8-45 and described in Table 8-55.

Return to the Summary Table.

GPIO C Pull Up Disable Register (GPIO64 to 95)
Disables the Pull-Up on GPIO.
0: Enables the Pull-Up.
1: Disables the Pull-Up.
Reading the register returns the current value of the register setting.

**Figure 8-45. GPCPUD Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO95 | GPIO94 | GPIO93 | GPIO92 | GPIO91 | GPIO90 | GPIO89 | GPIO88 |
| R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO87 | GPIO86 | GPIO85 | GPIO84 | GPIO83 | GPIO82 | GPIO81 | GPIO80 |
| R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO79 | GPIO78 | GPIO77 | GPIO76 | GPIO75 | GPIO74 | GPIO73 | GPIO72 |
| R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO71 | GPIO70 | GPIO69 | GPIO68 | GPIO67 | GPIO66 | GPIO65 | GPIO64 |
| R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h |

**Table 8-55. GPCPUD Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO95 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 30 | GPIO94 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 29 | GPIO93 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 28 | GPIO92 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 27 | GPIO91 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 26 | GPIO90 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 25 | GPIO89 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 24 | GPIO88 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 23 | GPIO87 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 22 | GPIO86 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 21 | GPIO85 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-55. GPCPUD Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 20 | GPIO84 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 19 | GPIO83 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 18 | GPIO82 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 17 | GPIO81 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 16 | GPIO80 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 15 | GPIO79 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 14 | GPIO78 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 13 | GPIO77 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 12 | GPIO76 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 11 | GPIO75 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 10 | GPIO74 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 9 | GPIO73 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 8 | GPIO72 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 7 | GPIO71 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 6 | GPIO70 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 5 | GPIO69 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 4 | GPIO68 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 3 | GPIO67 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 2 | GPIO66 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 1 | GPIO65 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 0 | GPIO64 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.43 GPCINV Register (Offset = 90h) [Reset = 00000000h]

GPCINV is shown in Figure 8-46 and described in Table 8-56.

Return to the Summary Table.

GPIO C Input Polarity Invert Registers (GPIO64 to 95)
Selects between non-inverted and inverted GPIO input to the device.
0: selects non-inverted GPIO input
1: selects inverted GPIO input
Reading the register returns the current value of the register setting.

**Figure 8-46. GPCINV Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO95 | GPIO94 | GPIO93 | GPIO92 | GPIO91 | GPIO90 | GPIO89 | GPIO88 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO87 | GPIO86 | GPIO85 | GPIO84 | GPIO83 | GPIO82 | GPIO81 | GPIO80 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO79 | GPIO78 | GPIO77 | GPIO76 | GPIO75 | GPIO74 | GPIO73 | GPIO72 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO71 | GPIO70 | GPIO69 | GPIO68 | GPIO67 | GPIO66 | GPIO65 | GPIO64 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 8-56. GPCINV Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO95 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 30 | GPIO94 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 29 | GPIO93 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 28 | GPIO92 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 27 | GPIO91 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 26 | GPIO90 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 25 | GPIO89 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 24 | GPIO88 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 23 | GPIO87 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 22 | GPIO86 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 21 | GPIO85 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-56. GPCINV Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 20 | GPIO84 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 19 | GPIO83 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 18 | GPIO82 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 17 | GPIO81 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 16 | GPIO80 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 15 | GPIO79 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 14 | GPIO78 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 13 | GPIO77 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 12 | GPIO76 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 11 | GPIO75 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 10 | GPIO74 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 9 | GPIO73 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 8 | GPIO72 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 7 | GPIO71 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 6 | GPIO70 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 5 | GPIO69 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 4 | GPIO68 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 3 | GPIO67 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 2 | GPIO66 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 1 | GPIO65 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 0 | GPIO64 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.44 GPCODR Register (Offset = 92h) [Reset = 00000000h]

GPCODR is shown in Figure 8-47 and described in Table 8-57.

Return to the Summary Table.

GPIO C Open Drain Output Register (GPIO64 to GPIO95)
Selects between normal and open-drain output for the GPIO pin.
0: Normal Output
1: Open Drain Output
Reading the register returns the current value of the register setting.
Note:
[1] In the Open Drain output mode, if the buffer is configured for output mode, a 0 value to be driven out comes out on the on the PAD while a 1 value to be driven out tri-states the buffer.

**Figure 8-47. GPCODR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO95 | GPIO94 | GPIO93 | GPIO92 | GPIO91 | GPIO90 | GPIO89 | GPIO88 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO87 | GPIO86 | GPIO85 | GPIO84 | GPIO83 | GPIO82 | GPIO81 | GPIO80 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO79 | GPIO78 | GPIO77 | GPIO76 | GPIO75 | GPIO74 | GPIO73 | GPIO72 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO71 | GPIO70 | GPIO69 | GPIO68 | GPIO67 | GPIO66 | GPIO65 | GPIO64 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 8-57. GPCODR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO95 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 30 | GPIO94 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 29 | GPIO93 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 28 | GPIO92 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 27 | GPIO91 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 26 | GPIO90 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 25 | GPIO89 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 24 | GPIO88 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 23 | GPIO87 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 22 | GPIO86 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-57. GPCODR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 21 | GPIO85 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 20 | GPIO84 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 19 | GPIO83 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 18 | GPIO82 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 17 | GPIO81 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 16 | GPIO80 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 15 | GPIO79 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 14 | GPIO78 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 13 | GPIO77 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 12 | GPIO76 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 11 | GPIO75 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 10 | GPIO74 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 9 | GPIO73 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 8 | GPIO72 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 7 | GPIO71 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 6 | GPIO70 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 5 | GPIO69 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 4 | GPIO68 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 3 | GPIO67 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 2 | GPIO66 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 1 | GPIO65 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 0 | GPIO64 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.45 GPCGMUX1 Register (Offset = A0h) [Reset = 00000000h]

GPCGMUX1 is shown in Figure 8-48 and described in Table 8-58.

Return to the Summary Table.

GPIO C Peripheral Group Mux (GPIO64 to 79)
Defines pin-muxing selection for GPIO.
Note: For complete pin-mux selection on GPIOx, GPAMUXy.GPIOx configuration is also required.

**Figure 8-48. GPCGMUX1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO79 | | GPIO78 | | GPIO77 | | GPIO76 | | GPIO75 | | GPIO74 | | GPIO73 | | GPIO72 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO71 | | GPIO70 | | GPIO69 | | GPIO68 | | GPIO67 | | GPIO66 | | GPIO65 | | GPIO64 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-58. GPCGMUX1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GPIO79 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 29-28 | GPIO78 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 27-26 | GPIO77 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 25-24 | GPIO76 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 23-22 | GPIO75 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 21-20 | GPIO74 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 19-18 | GPIO73 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 17-16 | GPIO72 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 15-14 | GPIO71 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 13-12 | GPIO70 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 11-10 | GPIO69 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 9-8 | GPIO68 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 7-6 | GPIO67 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 5-4 | GPIO66 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 3-2 | GPIO65 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 1-0 | GPIO64 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.46 GPCGMUX2 Register (Offset = A2h) [Reset = 00000000h]

GPCGMUX2 is shown in Figure 8-49 and described in Table 8-59.

Return to the Summary Table.

GPIO C Peripheral Group Mux (GPIO80 to 95)
Defines pin-muxing selection for GPIO.
Note: For complete pin-mux selection on GPIOx, GPAMUXy.GPIOx configuration is also required.

**Figure 8-49. GPCGMUX2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO95 | | GPIO94 | | GPIO93 | | GPIO92 | | GPIO91 | | GPIO90 | | GPIO89 | | GPIO88 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO87 | | GPIO86 | | GPIO85 | | GPIO84 | | GPIO83 | | GPIO82 | | GPIO81 | | GPIO80 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-59. GPCGMUX2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GPIO95 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 29-28 | GPIO94 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 27-26 | GPIO93 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 25-24 | GPIO92 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 23-22 | GPIO91 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 21-20 | GPIO90 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 19-18 | GPIO89 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 17-16 | GPIO88 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 15-14 | GPIO87 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 13-12 | GPIO86 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 11-10 | GPIO85 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 9-8 | GPIO84 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 7-6 | GPIO83 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 5-4 | GPIO82 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 3-2 | GPIO81 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 1-0 | GPIO80 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.47 GPCCSEL1 Register (Offset = A8h) [Reset = 00000000h]

GPCCSEL1 is shown in Figure 8-50 and described in Table 8-60.

Return to the Summary Table.

GPIO C Core Select Register (GPIO64 to 71)
Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
xx00: CPU1 selected
xx01: CPU1.CLA1 selected
xx10: CPU2 selected
xx11: CPU2.CLA1 selected

**Figure 8-50. GPCCSEL1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO71 | | | | GPIO70 | | | | GPIO69 | | | | GPIO68 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO67 | | | | GPIO66 | | | | GPIO65 | | | | GPIO64 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |

**Table 8-60. GPCCSEL1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-28 | GPIO71 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 27-24 | GPIO70 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 23-20 | GPIO69 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 19-16 | GPIO68 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 15-12 | GPIO67 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 11-8 | GPIO66 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 7-4 | GPIO65 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 3-0 | GPIO64 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.48 GPCCSEL2 Register (Offset = AAh) [Reset = 00000000h]

GPCCSEL2 is shown in Figure 8-51 and described in Table 8-61.

Return to the Summary Table.

GPIO C Core Select Register (GPIO72 to 79)
Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
xx00: CPU1 selected
xx01: CPU1.CLA1 selected
xx10: CPU2 selected
xx11: CPU2.CLA1 selected

**Figure 8-51. GPCCSEL2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO79 | | | | GPIO78 | | | | GPIO77 | | | | GPIO76 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO75 | | | | GPIO74 | | | | GPIO73 | | | | GPIO72 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |

**Table 8-61. GPCCSEL2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-28 | GPIO79 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 27-24 | GPIO78 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 23-20 | GPIO77 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 19-16 | GPIO76 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 15-12 | GPIO75 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 11-8 | GPIO74 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 7-4 | GPIO73 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 3-0 | GPIO72 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.49 GPCCSEL3 Register (Offset = ACh) [Reset = 00000000h]

GPCCSEL3 is shown in Figure 8-52 and described in Table 8-62.

Return to the Summary Table.

GPIO C Core Select Register (GPIO80 to 87)
Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
xx00: CPU1 selected
xx01: CPU1.CLA1 selected
xx10: CPU2 selected
xx11: CPU2.CLA1 selected

**Figure 8-52. GPCCSEL3 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO87 | | | | GPIO86 | | | | GPIO85 | | | | GPIO84 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO83 | | | | GPIO82 | | | | GPIO81 | | | | GPIO80 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |

**Table 8-62. GPCCSEL3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-28 | GPIO87 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 27-24 | GPIO86 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 23-20 | GPIO85 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 19-16 | GPIO84 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 15-12 | GPIO83 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 11-8 | GPIO82 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 7-4 | GPIO81 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 3-0 | GPIO80 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.50 GPCCSEL4 Register (Offset = AEh) [Reset = 00000000h]

GPCCSEL4 is shown in Figure 8-53 and described in Table 8-63.

Return to the Summary Table.

GPIO C Core Select Register (GPIO88 to 95)
Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
xx00: CPU1 selected
xx01: CPU1.CLA1 selected
xx10: CPU2 selected
xx11: CPU2.CLA1 selected

**Figure 8-53. GPCCSEL4 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO95 | | | | GPIO94 | | | | GPIO93 | | | | GPIO92 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO91 | | | | GPIO90 | | | | GPIO89 | | | | GPIO88 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |

**Table 8-63. GPCCSEL4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-28 | GPIO95 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 27-24 | GPIO94 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 23-20 | GPIO93 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 19-16 | GPIO92 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 15-12 | GPIO91 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 11-8 | GPIO90 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 7-4 | GPIO89 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 3-0 | GPIO88 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.51 GPCLOCK Register (Offset = BCh) [Reset = 00000000h]

GPCLOCK is shown in Figure 8-54 and described in Table 8-64.

Return to the Summary Table.

GPIO C Lock Configuration Register (GPIO64 to 95)
GPIO Configuration Lock for GPIO.
0: Bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx register which control the same pin can be changed
1: Locks changes to the bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx registers which control the same pin

**Figure 8-54. GPCLOCK Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO95 | GPIO94 | GPIO93 | GPIO92 | GPIO91 | GPIO90 | GPIO89 | GPIO88 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO87 | GPIO86 | GPIO85 | GPIO84 | GPIO83 | GPIO82 | GPIO81 | GPIO80 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO79 | GPIO78 | GPIO77 | GPIO76 | GPIO75 | GPIO74 | GPIO73 | GPIO72 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO71 | GPIO70 | GPIO69 | GPIO68 | GPIO67 | GPIO66 | GPIO65 | GPIO64 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 8-64. GPCLOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO95 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 30 | GPIO94 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 29 | GPIO93 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 28 | GPIO92 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 27 | GPIO91 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 26 | GPIO90 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 25 | GPIO89 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 24 | GPIO88 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 23 | GPIO87 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 22 | GPIO86 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 21 | GPIO85 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-64. GPCLOCK Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 20 | GPIO84 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 19 | GPIO83 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 18 | GPIO82 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 17 | GPIO81 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 16 | GPIO80 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 15 | GPIO79 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 14 | GPIO78 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 13 | GPIO77 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 12 | GPIO76 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 11 | GPIO75 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 10 | GPIO74 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 9 | GPIO73 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 8 | GPIO72 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 7 | GPIO71 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 6 | GPIO70 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 5 | GPIO69 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 4 | GPIO68 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 3 | GPIO67 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 2 | GPIO66 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 1 | GPIO65 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 0 | GPIO64 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.52 GPCCR Register (Offset = BEh) [Reset = 00000000h]

GPCCR is shown in Figure 8-55 and described in Table 8-65.

Return to the Summary Table.

GPIO C Lock Commit Register (GPIO64 to 95)
GPIO Configuration Lock Commit for GPIO:
1: Locks changes to the bit in GPyLOCK register which controls the same pin
0: Bit in the GPyLOCK register which controls the same pin can be changed

**Figure 8-55. GPCCR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO95 | GPIO94 | GPIO93 | GPIO92 | GPIO91 | GPIO90 | GPIO89 | GPIO88 |
| R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO87 | GPIO86 | GPIO85 | GPIO84 | GPIO83 | GPIO82 | GPIO81 | GPIO80 |
| R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO79 | GPIO78 | GPIO77 | GPIO76 | GPIO75 | GPIO74 | GPIO73 | GPIO72 |
| R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO71 | GPIO70 | GPIO69 | GPIO68 | GPIO67 | GPIO66 | GPIO65 | GPIO64 |
| R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h |

**Table 8-65. GPCCR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO95 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 30 | GPIO94 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 29 | GPIO93 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 28 | GPIO92 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 27 | GPIO91 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 26 | GPIO90 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 25 | GPIO89 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 24 | GPIO88 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 23 | GPIO87 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 22 | GPIO86 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 21 | GPIO85 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 20 | GPIO84 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-65. GPCCR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19 | GPIO83 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 18 | GPIO82 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 17 | GPIO81 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 16 | GPIO80 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 15 | GPIO79 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 14 | GPIO78 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 13 | GPIO77 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 12 | GPIO76 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 11 | GPIO75 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 10 | GPIO74 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 9 | GPIO73 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 8 | GPIO72 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 7 | GPIO71 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 6 | GPIO70 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 5 | GPIO69 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 4 | GPIO68 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 3 | GPIO67 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 2 | GPIO66 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 1 | GPIO65 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 0 | GPIO64 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.53 GPDCTRL Register (Offset = C0h) [Reset = 00000000h]

GPDCTRL is shown in Figure 8-56 and described in Table 8-66.

Return to the Summary Table.

GPIO D Qualification Sampling Period Control (GPIO96 to 127)

**Figure 8-56. GPDCTRL Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| QUALPRD3 | | | | | | | | QUALPRD2 | | | | | | | | QUALPRD1 | | | | | | | | QUALPRD0 | | | | | | | |
| R/W-0h | | | | | | | | R/W-0h | | | | | | | | R/W-0h | | | | | | | | R/W-0h | | | | | | | |

**Table 8-66. GPDCTRL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-24 | QUALPRD3 | R/W | 0h | Qualification sampling period for GPIO120 to GPIO127:<br>0x00,QUALPRDx = PLLSYSCLK<br>0x01,QUALPRDx = PLLSYSCLK/2<br>0x02,QUALPRDx = PLLSYSCLK/4<br>....<br>0xFF,QUALPRDx = PLLSYSCLK/510<br>Reset type: SYSRSn |
| 23-16 | QUALPRD2 | R/W | 0h | Qualification sampling period for GPIO112 to GPIO119:<br>0x00,QUALPRDx = PLLSYSCLK<br>0x01,QUALPRDx = PLLSYSCLK/2<br>0x02,QUALPRDx = PLLSYSCLK/4<br>....<br>0xFF,QUALPRDx = PLLSYSCLK/510<br>Reset type: SYSRSn |
| 15-8 | QUALPRD1 | R/W | 0h | Qualification sampling period for GPIO104 to GPIO111:<br>0x00,QUALPRDx = PLLSYSCLK<br>0x01,QUALPRDx = PLLSYSCLK/2<br>0x02,QUALPRDx = PLLSYSCLK/4<br>....<br>0xFF,QUALPRDx = PLLSYSCLK/510<br>Reset type: SYSRSn |
| 7-0 | QUALPRD0 | R/W | 0h | Qualification sampling period for GPIO96 to GPIO103:<br>0x00,QUALPRDx = PLLSYSCLK<br>0x01,QUALPRDx = PLLSYSCLK/2<br>0x02,QUALPRDx = PLLSYSCLK/4<br>....<br>0xFF,QUALPRDx = PLLSYSCLK/510<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.54 GPDQSEL1 Register (Offset = C2h) [Reset = 00000000h]

GPDQSEL1 is shown in Figure 8-57 and described in Table 8-67.

Return to the Summary Table.

GPIO D Qualifier Select 1 Register (GPIO96 to 111)
Input qualification type:
0,0 Sync
0,1 Qualification (3 samples)
1,0 Qualification (6 samples)
1,1 Async (no Sync or Qualification)

**Figure 8-57. GPDQSEL1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO111 | | GPIO110 | | GPIO109 | | GPIO108 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO107 | | GPIO106 | | GPIO105 | | GPIO104 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO103 | | GPIO102 | | GPIO101 | | GPIO100 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO99 | | GPIO98 | | GPIO97 | | GPIO96 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-67. GPDQSEL1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GPIO111 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 29-28 | GPIO110 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 27-26 | GPIO109 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 25-24 | GPIO108 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 23-22 | GPIO107 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 21-20 | GPIO106 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 19-18 | GPIO105 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 17-16 | GPIO104 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 15-14 | GPIO103 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 13-12 | GPIO102 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 11-10 | GPIO101 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-67. GPDQSEL1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 9-8 | GPIO100 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 7-6 | GPIO99 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 5-4 | GPIO98 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 3-2 | GPIO97 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 1-0 | GPIO96 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.55 GPDQSEL2 Register (Offset = C4h) [Reset = 00000000h]

GPDQSEL2 is shown in Figure 8-58 and described in Table 8-68.

Return to the Summary Table.

GPIO D Qualifier Select 2 Register (GPIO112 to 127)
Input qualification type:
0,0 Sync
0,1 Qualification (3 samples)
1,0 Qualification (6 samples)
1,1 Async (no Sync or Qualification)

**Figure 8-58. GPDQSEL2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO127 | | GPIO126 | | GPIO125 | | GPIO124 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO123 | | GPIO122 | | GPIO121 | | GPIO120 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO119 | | GPIO118 | | GPIO117 | | GPIO116 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO115 | | GPIO114 | | GPIO113 | | GPIO112 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-68. GPDQSEL2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GPIO127 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 29-28 | GPIO126 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 27-26 | GPIO125 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 25-24 | GPIO124 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 23-22 | GPIO123 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 21-20 | GPIO122 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 19-18 | GPIO121 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 17-16 | GPIO120 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 15-14 | GPIO119 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 13-12 | GPIO118 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 11-10 | GPIO117 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-68. GPDQSEL2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 9-8 | GPIO116 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 7-6 | GPIO115 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 5-4 | GPIO114 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 3-2 | GPIO113 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 1-0 | GPIO112 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.56 GPDMUX1 Register (Offset = C6h) [Reset = 00000000h]

GPDMUX1 is shown in Figure 8-59 and described in Table 8-69.

Return to the Summary Table.

GPIO D Mux 1 Register (GPIO96 to 111)
Defines pin-muxing selection for GPIO.
Notes:
The respective GPyGMUXn.GPIOz must be configured prior to this register to avoid intermediate peripheral selects being mapped to the GPIO.

**Figure 8-59. GPDMUX1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO111 | | GPIO110 | | GPIO109 | | GPIO108 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO107 | | GPIO106 | | GPIO105 | | GPIO104 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO103 | | GPIO102 | | GPIO101 | | GPIO100 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO99 | | GPIO98 | | GPIO97 | | GPIO96 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-69. GPDMUX1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GPIO111 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 29-28 | GPIO110 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 27-26 | GPIO109 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 25-24 | GPIO108 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 23-22 | GPIO107 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 21-20 | GPIO106 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 19-18 | GPIO105 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 17-16 | GPIO104 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 15-14 | GPIO103 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 13-12 | GPIO102 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 11-10 | GPIO101 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-69. GPDMUX1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 9-8 | GPIO100 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 7-6 | GPIO99 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 5-4 | GPIO98 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 3-2 | GPIO97 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 1-0 | GPIO96 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.57 GPDMUX2 Register (Offset = C8h) [Reset = 00000000h]

GPDMUX2 is shown in Figure 8-60 and described in Table 8-70.

Return to the Summary Table.

GPIO D Mux 2 Register (GPIO112 to 127)
Defines pin-muxing selection for GPIO.
Notes:
The respective GPyGMUXn.GPIOz must be configured prior to this register to avoid intermediate peripheral selects being mapped to the GPIO.

**Figure 8-60. GPDMUX2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO127 | | GPIO126 | | GPIO125 | | GPIO124 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO123 | | GPIO122 | | GPIO121 | | GPIO120 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO119 | | GPIO118 | | GPIO117 | | GPIO116 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO115 | | GPIO114 | | GPIO113 | | GPIO112 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-70. GPDMUX2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GPIO127 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 29-28 | GPIO126 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 27-26 | GPIO125 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 25-24 | GPIO124 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 23-22 | GPIO123 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 21-20 | GPIO122 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 19-18 | GPIO121 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 17-16 | GPIO120 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 15-14 | GPIO119 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 13-12 | GPIO118 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 11-10 | GPIO117 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-70. GPDMUX2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 9-8 | GPIO116 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 7-6 | GPIO115 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 5-4 | GPIO114 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 3-2 | GPIO113 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 1-0 | GPIO112 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.58 GPDDIR Register (Offset = CAh) [Reset = 00000000h]

GPDDIR is shown in Figure 8-61 and described in Table 8-71.

Return to the Summary Table.

GPIO D Direction Register (GPIO96 to 127)
Controls direction of GPIO pins when the specified pin is configured in GPIO mode.
0: Configures pin as input.
1: Configures pin as output.
Reading the register returns the current value of the register setting.

**Figure 8-61. GPDDIR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO127 | GPIO126 | GPIO125 | GPIO124 | GPIO123 | GPIO122 | GPIO121 | GPIO120 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO119 | GPIO118 | GPIO117 | GPIO116 | GPIO115 | GPIO114 | GPIO113 | GPIO112 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO111 | GPIO110 | GPIO109 | GPIO108 | GPIO107 | GPIO106 | GPIO105 | GPIO104 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO103 | GPIO102 | GPIO101 | GPIO100 | GPIO99 | GPIO98 | GPIO97 | GPIO96 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 8-71. GPDDIR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO127 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 30 | GPIO126 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 29 | GPIO125 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 28 | GPIO124 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 27 | GPIO123 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 26 | GPIO122 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 25 | GPIO121 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 24 | GPIO120 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 23 | GPIO119 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 22 | GPIO118 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 21 | GPIO117 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-71. GPDDIR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 20 | GPIO116 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 19 | GPIO115 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 18 | GPIO114 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 17 | GPIO113 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 16 | GPIO112 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 15 | GPIO111 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 14 | GPIO110 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 13 | GPIO109 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 12 | GPIO108 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 11 | GPIO107 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 10 | GPIO106 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 9 | GPIO105 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 8 | GPIO104 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 7 | GPIO103 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 6 | GPIO102 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 5 | GPIO101 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 4 | GPIO100 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 3 | GPIO99 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 2 | GPIO98 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 1 | GPIO97 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 0 | GPIO96 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.59 GPDPUD Register (Offset = CCh) [Reset = FFFFFFFFh]

GPDPUD is shown in Figure 8-62 and described in Table 8-72.

Return to the Summary Table.

GPIO D Pull Up Disable Register (GPIO96 to 127)
Disables the Pull-Up on GPIO.
0: Enables the Pull-Up.
1: Disables the Pull-Up.
Reading the register returns the current value of the register setting.

**Figure 8-62. GPDPUD Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO127 | GPIO126 | GPIO125 | GPIO124 | GPIO123 | GPIO122 | GPIO121 | GPIO120 |
| R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO119 | GPIO118 | GPIO117 | GPIO116 | GPIO115 | GPIO114 | GPIO113 | GPIO112 |
| R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO111 | GPIO110 | GPIO109 | GPIO108 | GPIO107 | GPIO106 | GPIO105 | GPIO104 |
| R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO103 | GPIO102 | GPIO101 | GPIO100 | GPIO99 | GPIO98 | GPIO97 | GPIO96 |
| R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h |

**Table 8-72. GPDPUD Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO127 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 30 | GPIO126 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 29 | GPIO125 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 28 | GPIO124 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 27 | GPIO123 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 26 | GPIO122 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 25 | GPIO121 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 24 | GPIO120 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 23 | GPIO119 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 22 | GPIO118 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 21 | GPIO117 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-72. GPDPUD Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 20 | GPIO116 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 19 | GPIO115 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 18 | GPIO114 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 17 | GPIO113 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 16 | GPIO112 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 15 | GPIO111 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 14 | GPIO110 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 13 | GPIO109 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 12 | GPIO108 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 11 | GPIO107 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 10 | GPIO106 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 9 | GPIO105 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 8 | GPIO104 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 7 | GPIO103 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 6 | GPIO102 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 5 | GPIO101 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 4 | GPIO100 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 3 | GPIO99 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 2 | GPIO98 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 1 | GPIO97 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 0 | GPIO96 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.60 GPDINV Register (Offset = D0h) [Reset = 00000000h]

GPDINV is shown in Figure 8-63 and described in Table 8-73.

Return to the Summary Table.

GPIO D Input Polarity Invert Registers (GPIO96 to 127)
Selects between non-inverted and inverted GPIO input to the device.
0: selects non-inverted GPIO input
1: selects inverted GPIO input
Reading the register returns the current value of the register setting.

**Figure 8-63. GPDINV Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO127 | GPIO126 | GPIO125 | GPIO124 | GPIO123 | GPIO122 | GPIO121 | GPIO120 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO119 | GPIO118 | GPIO117 | GPIO116 | GPIO115 | GPIO114 | GPIO113 | GPIO112 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO111 | GPIO110 | GPIO109 | GPIO108 | GPIO107 | GPIO106 | GPIO105 | GPIO104 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO103 | GPIO102 | GPIO101 | GPIO100 | GPIO99 | GPIO98 | GPIO97 | GPIO96 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 8-73. GPDINV Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO127 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 30 | GPIO126 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 29 | GPIO125 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 28 | GPIO124 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 27 | GPIO123 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 26 | GPIO122 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 25 | GPIO121 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 24 | GPIO120 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 23 | GPIO119 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 22 | GPIO118 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 21 | GPIO117 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-73. GPDINV Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 20 | GPIO116 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 19 | GPIO115 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 18 | GPIO114 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 17 | GPIO113 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 16 | GPIO112 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 15 | GPIO111 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 14 | GPIO110 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 13 | GPIO109 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 12 | GPIO108 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 11 | GPIO107 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 10 | GPIO106 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 9 | GPIO105 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 8 | GPIO104 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 7 | GPIO103 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 6 | GPIO102 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 5 | GPIO101 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 4 | GPIO100 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 3 | GPIO99 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 2 | GPIO98 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 1 | GPIO97 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 0 | GPIO96 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.61 GPDODR Register (Offset = D2h) [Reset = 00000000h]

GPDODR is shown in Figure 8-64 and described in Table 8-74.

Return to the Summary Table.

GPIO D Open Drain Output Register (GPIO96 to GPIO127)
Selects between normal and open-drain output for the GPIO pin.
0: Normal Output
1: Open Drain Output
Reading the register returns the current value of the register setting.
Note:
[1] In the Open Drain output mode, if the buffer is configured for output mode, a 0 value to be driven out comes out on the on the PAD while a 1 value to be driven out tri-states the buffer.

**Figure 8-64. GPDODR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO127 | GPIO126 | GPIO125 | GPIO124 | GPIO123 | GPIO122 | GPIO121 | GPIO120 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO119 | GPIO118 | GPIO117 | GPIO116 | GPIO115 | GPIO114 | GPIO113 | GPIO112 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO111 | GPIO110 | GPIO109 | GPIO108 | GPIO107 | GPIO106 | GPIO105 | GPIO104 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO103 | GPIO102 | GPIO101 | GPIO100 | GPIO99 | GPIO98 | GPIO97 | GPIO96 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 8-74. GPDODR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO127 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 30 | GPIO126 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 29 | GPIO125 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 28 | GPIO124 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 27 | GPIO123 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 26 | GPIO122 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 25 | GPIO121 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 24 | GPIO120 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 23 | GPIO119 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 22 | GPIO118 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-74. GPDODR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 21 | GPIO117 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 20 | GPIO116 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 19 | GPIO115 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 18 | GPIO114 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 17 | GPIO113 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 16 | GPIO112 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 15 | GPIO111 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 14 | GPIO110 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 13 | GPIO109 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 12 | GPIO108 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 11 | GPIO107 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 10 | GPIO106 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 9 | GPIO105 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 8 | GPIO104 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 7 | GPIO103 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 6 | GPIO102 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 5 | GPIO101 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 4 | GPIO100 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 3 | GPIO99 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 2 | GPIO98 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 1 | GPIO97 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 0 | GPIO96 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.62 GPDGMUX1 Register (Offset = E0h) [Reset = 00000000h]

GPDGMUX1 is shown in Figure 8-65 and described in Table 8-75.

Return to the Summary Table.

GPIO D Peripheral Group Mux (GPIO96 to 111)
Defines pin-muxing selection for GPIO.
Note: For complete pin-mux selection on GPIOx, GPAMUXy.GPIOx configuration is also required.

**Figure 8-65. GPDGMUX1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO111 | | GPIO110 | | GPIO109 | | GPIO108 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO107 | | GPIO106 | | GPIO105 | | GPIO104 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO103 | | GPIO102 | | GPIO101 | | GPIO100 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO99 | | GPIO98 | | GPIO97 | | GPIO96 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-75. GPDGMUX1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GPIO111 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 29-28 | GPIO110 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 27-26 | GPIO109 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 25-24 | GPIO108 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 23-22 | GPIO107 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 21-20 | GPIO106 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 19-18 | GPIO105 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 17-16 | GPIO104 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 15-14 | GPIO103 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 13-12 | GPIO102 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 11-10 | GPIO101 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 9-8 | GPIO100 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-75. GPDGMUX1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 7-6 | GPIO99 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 5-4 | GPIO98 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 3-2 | GPIO97 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 1-0 | GPIO96 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.63 GPDGMUX2 Register (Offset = E2h) [Reset = 00000000h]

GPDGMUX2 is shown in Figure 8-66 and described in Table 8-76.

Return to the Summary Table.

GPIO D Peripheral Group Mux (GPIO112 to 127)
Defines pin-muxing selection for GPIO.
Note: For complete pin-mux selection on GPIOx, GPAMUXy.GPIOx configuration is also required.

**Figure 8-66. GPDGMUX2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO127 | | GPIO126 | | GPIO125 | | GPIO124 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO123 | | GPIO122 | | GPIO121 | | GPIO120 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO119 | | GPIO118 | | GPIO117 | | GPIO116 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO115 | | GPIO114 | | GPIO113 | | GPIO112 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-76. GPDGMUX2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GPIO127 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 29-28 | GPIO126 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 27-26 | GPIO125 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 25-24 | GPIO124 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 23-22 | GPIO123 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 21-20 | GPIO122 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 19-18 | GPIO121 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 17-16 | GPIO120 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 15-14 | GPIO119 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 13-12 | GPIO118 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 11-10 | GPIO117 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 9-8 | GPIO116 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-76. GPDGMUX2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 7-6 | GPIO115 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 5-4 | GPIO114 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 3-2 | GPIO113 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 1-0 | GPIO112 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.64 GPDCSEL1 Register (Offset = E8h) [Reset = 00000000h]

GPDCSEL1 is shown in Figure 8-67 and described in Table 8-77.

Return to the Summary Table.

GPIO D Core Select Register (GPIO96 to 103)
Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
xx00: CPU1 selected
xx01: CPU1.CLA1 selected
xx10: CPU2 selected
xx11: CPU2.CLA1 selected

**Figure 8-67. GPDCSEL1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO103 | | | | GPIO102 | | | | GPIO101 | | | | GPIO100 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO99 | | | | GPIO98 | | | | GPIO97 | | | | GPIO96 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |

**Table 8-77. GPDCSEL1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-28 | GPIO103 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 27-24 | GPIO102 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 23-20 | GPIO101 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 19-16 | GPIO100 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 15-12 | GPIO99 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 11-8 | GPIO98 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 7-4 | GPIO97 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 3-0 | GPIO96 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.65 GPDCSEL2 Register (Offset = EAh) [Reset = 00000000h]

GPDCSEL2 is shown in Figure 8-68 and described in Table 8-78.

Return to the Summary Table.

GPIO D Core Select Register (GPIO104 to 111)
Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
xx00: CPU1 selected
xx01: CPU1.CLA1 selected
xx10: CPU2 selected
xx11: CPU2.CLA1 selected

**Figure 8-68. GPDCSEL2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO111 | | | | GPIO110 | | | | GPIO109 | | | | GPIO108 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO107 | | | | GPIO106 | | | | GPIO105 | | | | GPIO104 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |

**Table 8-78. GPDCSEL2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-28 | GPIO111 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 27-24 | GPIO110 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 23-20 | GPIO109 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 19-16 | GPIO108 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 15-12 | GPIO107 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 11-8 | GPIO106 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 7-4 | GPIO105 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 3-0 | GPIO104 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.66 GPDCSEL3 Register (Offset = ECh) [Reset = 00000000h]

GPDCSEL3 is shown in Figure 8-69 and described in Table 8-79.

Return to the Summary Table.

GPIO D Core Select Register (GPIO112 to 119)
Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
xx00: CPU1 selected
xx01: CPU1.CLA1 selected
xx10: CPU2 selected
xx11: CPU2.CLA1 selected

**Figure 8-69. GPDCSEL3 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO119 | | | | GPIO118 | | | | GPIO117 | | | | GPIO116 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO115 | | | | GPIO114 | | | | GPIO113 | | | | GPIO112 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |

**Table 8-79. GPDCSEL3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-28 | GPIO119 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 27-24 | GPIO118 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 23-20 | GPIO117 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 19-16 | GPIO116 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 15-12 | GPIO115 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 11-8 | GPIO114 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 7-4 | GPIO113 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 3-0 | GPIO112 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.67 GPDCSEL4 Register (Offset = EEh) [Reset = 00000000h]

GPDCSEL4 is shown in Figure 8-70 and described in Table 8-80.

Return to the Summary Table.

GPIO D Core Select Register (GPIO120 to 127)
Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
xx00: CPU1 selected
xx01: CPU1.CLA1 selected
xx10: CPU2 selected
xx11: CPU2.CLA1 selected

**Figure 8-70. GPDCSEL4 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO127 | | | | GPIO126 | | | | GPIO125 | | | | GPIO124 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO123 | | | | GPIO122 | | | | GPIO121 | | | | GPIO120 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |

**Table 8-80. GPDCSEL4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-28 | GPIO127 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 27-24 | GPIO126 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 23-20 | GPIO125 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 19-16 | GPIO124 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 15-12 | GPIO123 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 11-8 | GPIO122 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 7-4 | GPIO121 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 3-0 | GPIO120 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.68 GPDLOCK Register (Offset = FCh) [Reset = 00000000h]

GPDLOCK is shown in Figure 8-71 and described in Table 8-81.

Return to the Summary Table.

GPIO D Lock Configuration Register (GPIO96 to 127)
GPIO Configuration Lock for GPIO.
0: Bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx register which control the same pin can be changed
1: Locks changes to the bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx registers which control the same pin

**Figure 8-71. GPDLOCK Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO127 | GPIO126 | GPIO125 | GPIO124 | GPIO123 | GPIO122 | GPIO121 | GPIO120 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO119 | GPIO118 | GPIO117 | GPIO116 | GPIO115 | GPIO114 | GPIO113 | GPIO112 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO111 | GPIO110 | GPIO109 | GPIO108 | GPIO107 | GPIO106 | GPIO105 | GPIO104 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO103 | GPIO102 | GPIO101 | GPIO100 | GPIO99 | GPIO98 | GPIO97 | GPIO96 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 8-81. GPDLOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO127 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 30 | GPIO126 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 29 | GPIO125 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 28 | GPIO124 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 27 | GPIO123 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 26 | GPIO122 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 25 | GPIO121 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 24 | GPIO120 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 23 | GPIO119 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 22 | GPIO118 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 21 | GPIO117 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-81. GPDLOCK Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 20 | GPIO116 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 19 | GPIO115 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 18 | GPIO114 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 17 | GPIO113 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 16 | GPIO112 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 15 | GPIO111 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 14 | GPIO110 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 13 | GPIO109 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 12 | GPIO108 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 11 | GPIO107 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 10 | GPIO106 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 9 | GPIO105 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 8 | GPIO104 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 7 | GPIO103 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 6 | GPIO102 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 5 | GPIO101 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 4 | GPIO100 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 3 | GPIO99 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 2 | GPIO98 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 1 | GPIO97 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 0 | GPIO96 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.69 GPDCR Register (Offset = FEh) [Reset = 00000000h]

GPDCR is shown in Figure 8-72 and described in Table 8-82.

Return to the Summary Table.

GPIO D Lock Commit Register (GPIO96 to 127)
GPIO Configuration Lock Commit for GPIO:
1: Locks changes to the bit in GPyLOCK register which controls the same pin
0: Bit in the GPyLOCK register which controls the same pin can be changed

**Figure 8-72. GPDCR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO127 | GPIO126 | GPIO125 | GPIO124 | GPIO123 | GPIO122 | GPIO121 | GPIO120 |
| R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO119 | GPIO118 | GPIO117 | GPIO116 | GPIO115 | GPIO114 | GPIO113 | GPIO112 |
| R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO111 | GPIO110 | GPIO109 | GPIO108 | GPIO107 | GPIO106 | GPIO105 | GPIO104 |
| R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO103 | GPIO102 | GPIO101 | GPIO100 | GPIO99 | GPIO98 | GPIO97 | GPIO96 |
| R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h |

**Table 8-82. GPDCR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO127 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 30 | GPIO126 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 29 | GPIO125 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 28 | GPIO124 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 27 | GPIO123 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 26 | GPIO122 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 25 | GPIO121 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 24 | GPIO120 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 23 | GPIO119 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 22 | GPIO118 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 21 | GPIO117 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 20 | GPIO116 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-82. GPDCR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19 | GPIO115 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 18 | GPIO114 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 17 | GPIO113 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 16 | GPIO112 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 15 | GPIO111 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 14 | GPIO110 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 13 | GPIO109 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 12 | GPIO108 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 11 | GPIO107 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 10 | GPIO106 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 9 | GPIO105 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 8 | GPIO104 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 7 | GPIO103 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 6 | GPIO102 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 5 | GPIO101 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 4 | GPIO100 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 3 | GPIO99 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 2 | GPIO98 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 1 | GPIO97 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 0 | GPIO96 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.70 GPECTRL Register (Offset = 100h) [Reset = 00000000h]

GPECTRL is shown in Figure 8-73 and described in Table 8-83.

Return to the Summary Table.

GPIO E Qualification Sampling Period Control (GPIO128 to 159)

**Figure 8-73. GPECTRL Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| QUALPRD3 | | | | | | | | QUALPRD2 | | | | | | | | QUALPRD1 | | | | | | | | QUALPRD0 | | | | | | | |
| R/W-0h | | | | | | | | R/W-0h | | | | | | | | R/W-0h | | | | | | | | R/W-0h | | | | | | | |

**Table 8-83. GPECTRL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-24 | QUALPRD3 | R/W | 0h | Qualification sampling period for GPIO152 to GPIO159:<br>0x00,QUALPRDx = PLLSYSCLK<br>0x01,QUALPRDx = PLLSYSCLK/2<br>0x02,QUALPRDx = PLLSYSCLK/4<br>....<br>0xFF,QUALPRDx = PLLSYSCLK/510<br>Reset type: SYSRSn |
| 23-16 | QUALPRD2 | R/W | 0h | Qualification sampling period for GPIO144 to GPIO151:<br>0x00,QUALPRDx = PLLSYSCLK<br>0x01,QUALPRDx = PLLSYSCLK/2<br>0x02,QUALPRDx = PLLSYSCLK/4<br>....<br>0xFF,QUALPRDx = PLLSYSCLK/510<br>Reset type: SYSRSn |
| 15-8 | QUALPRD1 | R/W | 0h | Qualification sampling period for GPIO136 to GPIO143:<br>0x00,QUALPRDx = PLLSYSCLK<br>0x01,QUALPRDx = PLLSYSCLK/2<br>0x02,QUALPRDx = PLLSYSCLK/4<br>....<br>0xFF,QUALPRDx = PLLSYSCLK/510<br>Reset type: SYSRSn |
| 7-0 | QUALPRD0 | R/W | 0h | Qualification sampling period for GPIO128 to GPIO135:<br>0x00,QUALPRDx = PLLSYSCLK<br>0x01,QUALPRDx = PLLSYSCLK/2<br>0x02,QUALPRDx = PLLSYSCLK/4<br>....<br>0xFF,QUALPRDx = PLLSYSCLK/510<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.71 GPEQSEL1 Register (Offset = 102h) [Reset = 00000000h]

GPEQSEL1 is shown in Figure 8-74 and described in Table 8-84.

Return to the Summary Table.

GPIO E Qualifier Select 1 Register (GPIO128 to 143)
Input qualification type:
0,0 Sync
0,1 Qualification (3 samples)
1,0 Qualification (6 samples)
1,1 Async (no Sync or Qualification)

**Figure 8-74. GPEQSEL1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO143 | | GPIO142 | | GPIO141 | | GPIO140 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO139 | | GPIO138 | | GPIO137 | | GPIO136 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO135 | | GPIO134 | | GPIO133 | | GPIO132 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO131 | | GPIO130 | | GPIO129 | | GPIO128 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-84. GPEQSEL1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GPIO143 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 29-28 | GPIO142 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 27-26 | GPIO141 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 25-24 | GPIO140 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 23-22 | GPIO139 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 21-20 | GPIO138 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 19-18 | GPIO137 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 17-16 | GPIO136 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 15-14 | GPIO135 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 13-12 | GPIO134 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 11-10 | GPIO133 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-84. GPEQSEL1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 9-8 | GPIO132 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 7-6 | GPIO131 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 5-4 | GPIO130 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 3-2 | GPIO129 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 1-0 | GPIO128 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.72 GPEQSEL2 Register (Offset = 104h) [Reset = 00000000h]

GPEQSEL2 is shown in Figure 8-75 and described in Table 8-85.

Return to the Summary Table.

GPIO E Qualifier Select 2 Register (GPIO144 to 159)
Input qualification type:
0,0 Sync
0,1 Qualification (3 samples)
1,0 Qualification (6 samples)
1,1 Async (no Sync or Qualification)

**Figure 8-75. GPEQSEL2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO159 | | GPIO158 | | GPIO157 | | GPIO156 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO155 | | GPIO154 | | GPIO153 | | GPIO152 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO151 | | GPIO150 | | GPIO149 | | GPIO148 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO147 | | GPIO146 | | GPIO145 | | GPIO144 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-85. GPEQSEL2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GPIO159 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 29-28 | GPIO158 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 27-26 | GPIO157 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 25-24 | GPIO156 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 23-22 | GPIO155 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 21-20 | GPIO154 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 19-18 | GPIO153 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 17-16 | GPIO152 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 15-14 | GPIO151 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 13-12 | GPIO150 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 11-10 | GPIO149 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-85. GPEQSEL2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 9-8 | GPIO148 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 7-6 | GPIO147 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 5-4 | GPIO146 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 3-2 | GPIO145 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 1-0 | GPIO144 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.73 GPEMUX1 Register (Offset = 106h) [Reset = 00000000h]

GPEMUX1 is shown in Figure 8-76 and described in Table 8-86.

Return to the Summary Table.

GPIO E Mux 1 Register (GPIO128 to 143)
Defines pin-muxing selection for GPIO.
Notes:
The respective GPyGMUXn.GPIOz must be configured prior to this register to avoid intermediate peripheral selects being mapped to the GPIO.

**Figure 8-76. GPEMUX1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO143 | | GPIO142 | | GPIO141 | | GPIO140 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO139 | | GPIO138 | | GPIO137 | | GPIO136 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO135 | | GPIO134 | | GPIO133 | | GPIO132 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO131 | | GPIO130 | | GPIO129 | | GPIO128 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-86. GPEMUX1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GPIO143 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 29-28 | GPIO142 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 27-26 | GPIO141 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 25-24 | GPIO140 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 23-22 | GPIO139 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 21-20 | GPIO138 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 19-18 | GPIO137 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 17-16 | GPIO136 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 15-14 | GPIO135 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 13-12 | GPIO134 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 11-10 | GPIO133 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-86. GPEMUX1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 9-8 | GPIO132 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 7-6 | GPIO131 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 5-4 | GPIO130 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 3-2 | GPIO129 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 1-0 | GPIO128 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.74 GPEMUX2 Register (Offset = 108h) [Reset = 00000000h]

GPEMUX2 is shown in Figure 8-77 and described in Table 8-87.

Return to the Summary Table.

GPIO E Mux 2 Register (GPIO144 to 159)
Defines pin-muxing selection for GPIO.
Notes:
The respective GPyGMUXn.GPIOz must be configured prior to this register to avoid intermediate peripheral selects being mapped to the GPIO.

**Figure 8-77. GPEMUX2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO159 | | GPIO158 | | GPIO157 | | GPIO156 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO155 | | GPIO154 | | GPIO153 | | GPIO152 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO151 | | GPIO150 | | GPIO149 | | GPIO148 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO147 | | GPIO146 | | GPIO145 | | GPIO144 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-87. GPEMUX2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GPIO159 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 29-28 | GPIO158 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 27-26 | GPIO157 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 25-24 | GPIO156 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 23-22 | GPIO155 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 21-20 | GPIO154 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 19-18 | GPIO153 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 17-16 | GPIO152 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 15-14 | GPIO151 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 13-12 | GPIO150 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 11-10 | GPIO149 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-87. GPEMUX2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 9-8 | GPIO148 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 7-6 | GPIO147 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 5-4 | GPIO146 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 3-2 | GPIO145 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 1-0 | GPIO144 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.75 GPEDIR Register (Offset = 10Ah) [Reset = 00000000h]

GPEDIR is shown in Figure 8-78 and described in Table 8-88.

Return to the Summary Table.

GPIO E Direction Register (GPIO128 to 159)
Controls direction of GPIO pins when the specified pin is configured in GPIO mode.
0: Configures pin as input.
1: Configures pin as output.
Reading the register returns the current value of the register setting.

**Figure 8-78. GPEDIR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO159 | GPIO158 | GPIO157 | GPIO156 | GPIO155 | GPIO154 | GPIO153 | GPIO152 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO151 | GPIO150 | GPIO149 | GPIO148 | GPIO147 | GPIO146 | GPIO145 | GPIO144 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO143 | GPIO142 | GPIO141 | GPIO140 | GPIO139 | GPIO138 | GPIO137 | GPIO136 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO135 | GPIO134 | GPIO133 | GPIO132 | GPIO131 | GPIO130 | GPIO129 | GPIO128 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 8-88. GPEDIR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO159 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 30 | GPIO158 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 29 | GPIO157 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 28 | GPIO156 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 27 | GPIO155 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 26 | GPIO154 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 25 | GPIO153 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 24 | GPIO152 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 23 | GPIO151 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 22 | GPIO150 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 21 | GPIO149 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-88. GPEDIR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 20 | GPIO148 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 19 | GPIO147 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 18 | GPIO146 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 17 | GPIO145 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 16 | GPIO144 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 15 | GPIO143 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 14 | GPIO142 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 13 | GPIO141 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 12 | GPIO140 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 11 | GPIO139 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 10 | GPIO138 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 9 | GPIO137 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 8 | GPIO136 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 7 | GPIO135 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 6 | GPIO134 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 5 | GPIO133 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 4 | GPIO132 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 3 | GPIO131 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 2 | GPIO130 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 1 | GPIO129 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 0 | GPIO128 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.76 GPEPUD Register (Offset = 10Ch) [Reset = FFFFFFFFh]

GPEPUD is shown in Figure 8-79 and described in Table 8-89.

Return to the Summary Table.

GPIO E Pull Up Disable Register (GPIO128 to 159)
Disables the Pull-Up on GPIO.
0: Enables the Pull-Up.
1: Disables the Pull-Up.
Reading the register returns the current value of the register setting.

**Figure 8-79. GPEPUD Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO159 | GPIO158 | GPIO157 | GPIO156 | GPIO155 | GPIO154 | GPIO153 | GPIO152 |
| R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO151 | GPIO150 | GPIO149 | GPIO148 | GPIO147 | GPIO146 | GPIO145 | GPIO144 |
| R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO143 | GPIO142 | GPIO141 | GPIO140 | GPIO139 | GPIO138 | GPIO137 | GPIO136 |
| R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO135 | GPIO134 | GPIO133 | GPIO132 | GPIO131 | GPIO130 | GPIO129 | GPIO128 |
| R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h |

**Table 8-89. GPEPUD Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO159 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 30 | GPIO158 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 29 | GPIO157 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 28 | GPIO156 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 27 | GPIO155 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 26 | GPIO154 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 25 | GPIO153 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 24 | GPIO152 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 23 | GPIO151 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 22 | GPIO150 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 21 | GPIO149 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-89. GPEPUD Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 20 | GPIO148 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 19 | GPIO147 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 18 | GPIO146 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 17 | GPIO145 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 16 | GPIO144 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 15 | GPIO143 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 14 | GPIO142 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 13 | GPIO141 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 12 | GPIO140 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 11 | GPIO139 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 10 | GPIO138 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 9 | GPIO137 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 8 | GPIO136 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 7 | GPIO135 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 6 | GPIO134 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 5 | GPIO133 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 4 | GPIO132 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 3 | GPIO131 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 2 | GPIO130 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 1 | GPIO129 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 0 | GPIO128 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.77 GPEINV Register (Offset = 110h) [Reset = 00000000h]

GPEINV is shown in Figure 8-80 and described in Table 8-90.

Return to the Summary Table.

GPIO E Input Polarity Invert Registers (GPIO128 to 159)
Selects between non-inverted and inverted GPIO input to the device.
0: selects non-inverted GPIO input
1: selects inverted GPIO input
Reading the register returns the current value of the register setting.

**Figure 8-80. GPEINV Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO159 | GPIO158 | GPIO157 | GPIO156 | GPIO155 | GPIO154 | GPIO153 | GPIO152 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO151 | GPIO150 | GPIO149 | GPIO148 | GPIO147 | GPIO146 | GPIO145 | GPIO144 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO143 | GPIO142 | GPIO141 | GPIO140 | GPIO139 | GPIO138 | GPIO137 | GPIO136 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO135 | GPIO134 | GPIO133 | GPIO132 | GPIO131 | GPIO130 | GPIO129 | GPIO128 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 8-90. GPEINV Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO159 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 30 | GPIO158 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 29 | GPIO157 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 28 | GPIO156 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 27 | GPIO155 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 26 | GPIO154 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 25 | GPIO153 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 24 | GPIO152 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 23 | GPIO151 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 22 | GPIO150 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 21 | GPIO149 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-90. GPEINV Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 20 | GPIO148 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 19 | GPIO147 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 18 | GPIO146 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 17 | GPIO145 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 16 | GPIO144 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 15 | GPIO143 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 14 | GPIO142 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 13 | GPIO141 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 12 | GPIO140 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 11 | GPIO139 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 10 | GPIO138 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 9 | GPIO137 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 8 | GPIO136 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 7 | GPIO135 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 6 | GPIO134 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 5 | GPIO133 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 4 | GPIO132 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 3 | GPIO131 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 2 | GPIO130 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 1 | GPIO129 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 0 | GPIO128 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.78 GPEODR Register (Offset = 112h) [Reset = 00000000h]

GPEODR is shown in Figure 8-81 and described in Table 8-91.

Return to the Summary Table.

GPIO E Open Drain Output Register (GPIO128 to GPIO159)
Selects between normal and open-drain output for the GPIO pin.
0: Normal Output
1: Open Drain Output
Reading the register returns the current value of the register setting.
Note:
[1] In the Open Drain output mode, if the buffer is configured for output mode, a 0 value to be driven out comes out on the on the PAD while a 1 value to be driven out tri-states the buffer.

**Figure 8-81. GPEODR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO159 | GPIO158 | GPIO157 | GPIO156 | GPIO155 | GPIO154 | GPIO153 | GPIO152 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO151 | GPIO150 | GPIO149 | GPIO148 | GPIO147 | GPIO146 | GPIO145 | GPIO144 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO143 | GPIO142 | GPIO141 | GPIO140 | GPIO139 | GPIO138 | GPIO137 | GPIO136 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO135 | GPIO134 | GPIO133 | GPIO132 | GPIO131 | GPIO130 | GPIO129 | GPIO128 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 8-91. GPEODR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO159 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 30 | GPIO158 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 29 | GPIO157 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 28 | GPIO156 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 27 | GPIO155 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 26 | GPIO154 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 25 | GPIO153 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 24 | GPIO152 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 23 | GPIO151 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 22 | GPIO150 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-91. GPEODR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 21 | GPIO149 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 20 | GPIO148 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 19 | GPIO147 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 18 | GPIO146 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 17 | GPIO145 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 16 | GPIO144 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 15 | GPIO143 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 14 | GPIO142 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 13 | GPIO141 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 12 | GPIO140 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 11 | GPIO139 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 10 | GPIO138 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 9 | GPIO137 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 8 | GPIO136 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 7 | GPIO135 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 6 | GPIO134 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 5 | GPIO133 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 4 | GPIO132 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 3 | GPIO131 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 2 | GPIO130 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 1 | GPIO129 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 0 | GPIO128 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.79 GPEGMUX1 Register (Offset = 120h) [Reset = 00000000h]

GPEGMUX1 is shown in Figure 8-82 and described in Table 8-92.

Return to the Summary Table.

GPIO E Peripheral Group Mux (GPIO128 to 143)
Defines pin-muxing selection for GPIO.
Note: For complete pin-mux selection on GPIOx, GPAMUXy.GPIOx configuration is also required.

**Figure 8-82. GPEGMUX1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO143 | | GPIO142 | | GPIO141 | | GPIO140 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO139 | | GPIO138 | | GPIO137 | | GPIO136 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO135 | | GPIO134 | | GPIO133 | | GPIO132 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO131 | | GPIO130 | | GPIO129 | | GPIO128 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-92. GPEGMUX1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GPIO143 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 29-28 | GPIO142 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 27-26 | GPIO141 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 25-24 | GPIO140 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 23-22 | GPIO139 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 21-20 | GPIO138 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 19-18 | GPIO137 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 17-16 | GPIO136 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 15-14 | GPIO135 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 13-12 | GPIO134 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 11-10 | GPIO133 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 9-8 | GPIO132 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-92. GPEGMUX1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 7-6 | GPIO131 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 5-4 | GPIO130 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 3-2 | GPIO129 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 1-0 | GPIO128 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.80 GPEGMUX2 Register (Offset = 122h) [Reset = 00000000h]

GPEGMUX2 is shown in Figure 8-83 and described in Table 8-93.

Return to the Summary Table.

GPIO E Peripheral Group Mux (GPIO144 to 159)
Defines pin-muxing selection for GPIO.
Note: For complete pin-mux selection on GPIOx, GPAMUXy.GPIOx configuration is also required.

**Figure 8-83. GPEGMUX2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO159 | | GPIO158 | | GPIO157 | | GPIO156 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO155 | | GPIO154 | | GPIO153 | | GPIO152 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO151 | | GPIO150 | | GPIO149 | | GPIO148 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO147 | | GPIO146 | | GPIO145 | | GPIO144 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-93. GPEGMUX2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GPIO159 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 29-28 | GPIO158 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 27-26 | GPIO157 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 25-24 | GPIO156 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 23-22 | GPIO155 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 21-20 | GPIO154 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 19-18 | GPIO153 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 17-16 | GPIO152 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 15-14 | GPIO151 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 13-12 | GPIO150 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 11-10 | GPIO149 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 9-8 | GPIO148 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-93. GPEGMUX2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 7-6 | GPIO147 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 5-4 | GPIO146 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 3-2 | GPIO145 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 1-0 | GPIO144 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.81 GPECSEL1 Register (Offset = 128h) [Reset = 00000000h]

GPECSEL1 is shown in Figure 8-84 and described in Table 8-94.

Return to the Summary Table.

GPIO E Core Select Register (GPIO128 to 135)
Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
xx00: CPU1 selected
xx01: CPU1.CLA1 selected
xx10: CPU2 selected
xx11: CPU2.CLA1 selected

**Figure 8-84. GPECSEL1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO135 | | | | GPIO134 | | | | GPIO133 | | | | GPIO132 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO131 | | | | GPIO130 | | | | GPIO129 | | | | GPIO128 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |

**Table 8-94. GPECSEL1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-28 | GPIO135 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 27-24 | GPIO134 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 23-20 | GPIO133 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 19-16 | GPIO132 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 15-12 | GPIO131 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 11-8 | GPIO130 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 7-4 | GPIO129 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 3-0 | GPIO128 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.82 GPECSEL2 Register (Offset = 12Ah) [Reset = 00000000h]

GPECSEL2 is shown in Figure 8-85 and described in Table 8-95.

Return to the Summary Table.

GPIO E Core Select Register (GPIO136 to 143)
Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
xx00: CPU1 selected
xx01: CPU1.CLA1 selected
xx10: CPU2 selected
xx11: CPU2.CLA1 selected

**Figure 8-85. GPECSEL2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO143 | | | | GPIO142 | | | | GPIO141 | | | | GPIO140 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO139 | | | | GPIO138 | | | | GPIO137 | | | | GPIO136 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |

**Table 8-95. GPECSEL2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-28 | GPIO143 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 27-24 | GPIO142 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 23-20 | GPIO141 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 19-16 | GPIO140 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 15-12 | GPIO139 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 11-8 | GPIO138 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 7-4 | GPIO137 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 3-0 | GPIO136 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.83 GPECSEL3 Register (Offset = 12Ch) [Reset = 00000000h]

GPECSEL3 is shown in Figure 8-86 and described in Table 8-96.

Return to the Summary Table.

GPIO E Core Select Register (GPIO144 to 151)
Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
xx00: CPU1 selected
xx01: CPU1.CLA1 selected
xx10: CPU2 selected
xx11: CPU2.CLA1 selected

**Figure 8-86. GPECSEL3 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO151 | | | | GPIO150 | | | | GPIO149 | | | | GPIO148 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO147 | | | | GPIO146 | | | | GPIO145 | | | | GPIO144 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |

**Table 8-96. GPECSEL3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-28 | GPIO151 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 27-24 | GPIO150 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 23-20 | GPIO149 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 19-16 | GPIO148 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 15-12 | GPIO147 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 11-8 | GPIO146 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 7-4 | GPIO145 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 3-0 | GPIO144 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.84 GPECSEL4 Register (Offset = 12Eh) [Reset = 00000000h]

GPECSEL4 is shown in Figure 8-87 and described in Table 8-97.

Return to the Summary Table.

GPIO E Core Select Register (GPIO152 to 159)
Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
xx00: CPU1 selected
xx01: CPU1.CLA1 selected
xx10: CPU2 selected
xx11: CPU2.CLA1 selected

**Figure 8-87. GPECSEL4 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO159 | | | | GPIO158 | | | | GPIO157 | | | | GPIO156 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO155 | | | | GPIO154 | | | | GPIO153 | | | | GPIO152 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |

**Table 8-97. GPECSEL4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-28 | GPIO159 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 27-24 | GPIO158 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 23-20 | GPIO157 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 19-16 | GPIO156 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 15-12 | GPIO155 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 11-8 | GPIO154 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 7-4 | GPIO153 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 3-0 | GPIO152 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.85 GPELOCK Register (Offset = 13Ch) [Reset = 00000000h]

GPELOCK is shown in Figure 8-88 and described in Table 8-98.

Return to the Summary Table.

GPIO E Lock Configuration Register (GPIO128 to 159)
GPIO Configuration Lock for GPIO.
0: Bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx register which control the same pin can be changed
1: Locks changes to the bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx registers which control the same pin

**Figure 8-88. GPELOCK Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO159 | GPIO158 | GPIO157 | GPIO156 | GPIO155 | GPIO154 | GPIO153 | GPIO152 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO151 | GPIO150 | GPIO149 | GPIO148 | GPIO147 | GPIO146 | GPIO145 | GPIO144 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO143 | GPIO142 | GPIO141 | GPIO140 | GPIO139 | GPIO138 | GPIO137 | GPIO136 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO135 | GPIO134 | GPIO133 | GPIO132 | GPIO131 | GPIO130 | GPIO129 | GPIO128 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 8-98. GPELOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO159 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 30 | GPIO158 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 29 | GPIO157 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 28 | GPIO156 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 27 | GPIO155 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 26 | GPIO154 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 25 | GPIO153 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 24 | GPIO152 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 23 | GPIO151 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 22 | GPIO150 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 21 | GPIO149 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-98. GPELOCK Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 20 | GPIO148 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 19 | GPIO147 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 18 | GPIO146 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 17 | GPIO145 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 16 | GPIO144 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 15 | GPIO143 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 14 | GPIO142 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 13 | GPIO141 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 12 | GPIO140 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 11 | GPIO139 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 10 | GPIO138 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 9 | GPIO137 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 8 | GPIO136 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 7 | GPIO135 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 6 | GPIO134 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 5 | GPIO133 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 4 | GPIO132 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 3 | GPIO131 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 2 | GPIO130 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 1 | GPIO129 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 0 | GPIO128 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.86 GPECR Register (Offset = 13Eh) [Reset = 00000000h]

GPECR is shown in Figure 8-89 and described in Table 8-99.

Return to the Summary Table.

GPIO E Lock Commit Register (GPIO128 to 159)
GPIO Configuration Lock Commit for GPIO:
1: Locks changes to the bit in GPyLOCK register which controls the same pin
0: Bit in the GPyLOCK register which controls the same pin can be changed

**Figure 8-89. GPECR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO159 | GPIO158 | GPIO157 | GPIO156 | GPIO155 | GPIO154 | GPIO153 | GPIO152 |
| R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO151 | GPIO150 | GPIO149 | GPIO148 | GPIO147 | GPIO146 | GPIO145 | GPIO144 |
| R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO143 | GPIO142 | GPIO141 | GPIO140 | GPIO139 | GPIO138 | GPIO137 | GPIO136 |
| R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO135 | GPIO134 | GPIO133 | GPIO132 | GPIO131 | GPIO130 | GPIO129 | GPIO128 |
| R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h |

**Table 8-99. GPECR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO159 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 30 | GPIO158 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 29 | GPIO157 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 28 | GPIO156 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 27 | GPIO155 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 26 | GPIO154 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 25 | GPIO153 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 24 | GPIO152 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 23 | GPIO151 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 22 | GPIO150 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 21 | GPIO149 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 20 | GPIO148 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-99. GPECR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19 | GPIO147 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 18 | GPIO146 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 17 | GPIO145 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 16 | GPIO144 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 15 | GPIO143 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 14 | GPIO142 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 13 | GPIO141 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 12 | GPIO140 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 11 | GPIO139 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 10 | GPIO138 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 9 | GPIO137 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 8 | GPIO136 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 7 | GPIO135 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 6 | GPIO134 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 5 | GPIO133 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 4 | GPIO132 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 3 | GPIO131 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 2 | GPIO130 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 1 | GPIO129 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 0 | GPIO128 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.87 GPFCTRL Register (Offset = 140h) [Reset = 00000000h]

GPFCTRL is shown in Figure 8-90 and described in Table 8-100.

Return to the Summary Table.

GPIO F Qualification Sampling Period Control (GPIO160 to 168)

**Figure 8-90. GPFCTRL Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | RESERVED | | | | | | | | QUALPRD1 | | | | | | | | QUALPRD0 | | | | | | | |
| R/W-0h | | | | | | | | R/W-0h | | | | | | | | R/W-0h | | | | | | | | R/W-0h | | | | | | | |

**Table 8-100. GPFCTRL 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 GPIO168:<br>0x00,QUALPRDx = PLLSYSCLK<br>0x01,QUALPRDx = PLLSYSCLK/2<br>0x02,QUALPRDx = PLLSYSCLK/4<br>....<br>0xFF,QUALPRDx = PLLSYSCLK/510<br>Reset type: SYSRSn |
| 7-0 | QUALPRD0 | R/W | 0h | Qualification sampling period for GPIO160 to GPIO167:<br>0x00,QUALPRDx = PLLSYSCLK<br>0x01,QUALPRDx = PLLSYSCLK/2<br>0x02,QUALPRDx = PLLSYSCLK/4<br>....<br>0xFF,QUALPRDx = PLLSYSCLK/510<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.88 GPFQSEL1 Register (Offset = 142h) [Reset = 00000000h]

GPFQSEL1 is shown in Figure 8-91 and described in Table 8-101.

Return to the Summary Table.

GPIO F Qualifier Select 1 Register (GPIO160 to 168)
Input qualification type:
0,0 Sync
0,1 Qualification (3 samples)
1,0 Qualification (6 samples)
1,1 Async (no Sync or Qualification)

**Figure 8-91. GPFQSEL1 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 | | GPIO168 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO167 | | GPIO166 | | GPIO165 | | GPIO164 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO163 | | GPIO162 | | GPIO161 | | GPIO160 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-101. GPFQSEL1 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 | GPIO168 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 15-14 | GPIO167 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 13-12 | GPIO166 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 11-10 | GPIO165 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 9-8 | GPIO164 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 7-6 | GPIO163 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 5-4 | GPIO162 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-101. GPFQSEL1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-2 | GPIO161 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |
| 1-0 | GPIO160 | R/W | 0h | Input qualification type<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.89 GPFMUX1 Register (Offset = 146h) [Reset = 00000000h]

GPFMUX1 is shown in Figure 8-92 and described in Table 8-102.

Return to the Summary Table.

GPIO F Mux 1 Register (GPIO160 to 168)
Defines pin-muxing selection for GPIO.
Notes:
The respective GPyGMUXn.GPIOz must be configured prior to this register to avoid intermediate peripheral selects being mapped to the GPIO.

**Figure 8-92. GPFMUX1 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 | | GPIO168 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO167 | | GPIO166 | | GPIO165 | | GPIO164 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO163 | | GPIO162 | | GPIO161 | | GPIO160 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-102. GPFMUX1 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 | GPIO168 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 15-14 | GPIO167 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 13-12 | GPIO166 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 11-10 | GPIO165 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 9-8 | GPIO164 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 7-6 | GPIO163 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 5-4 | GPIO162 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 3-2 | GPIO161 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-102. GPFMUX1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1-0 | GPIO160 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.90 GPFDIR Register (Offset = 14Ah) [Reset = 00000000h]

GPFDIR is shown in Figure 8-93 and described in Table 8-103.

Return to the Summary Table.

GPIO F Direction Register (GPIO160 to 168)
Controls direction of GPIO pins when the specified pin is configured in GPIO mode.
0: Configures pin as input.
1: Configures pin as output.
Reading the register returns the current value of the register setting.

**Figure 8-93. GPFDIR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | 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 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | 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 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | GPIO168 |
| 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 |
| GPIO167 | GPIO166 | GPIO165 | GPIO164 | GPIO163 | GPIO162 | GPIO161 | GPIO160 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 8-103. GPFDIR 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 |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-103. GPFDIR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 9 | RESERVED | R/W | 0h | Reserved |
| 8 | GPIO168 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 7 | GPIO167 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 6 | GPIO166 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 5 | GPIO165 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 4 | GPIO164 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 3 | GPIO163 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 2 | GPIO162 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 1 | GPIO161 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |
| 0 | GPIO160 | R/W | 0h | Defines direction for this pin in GPIO mode<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.91 GPFPUD Register (Offset = 14Ch) [Reset = FFFFFFFFh]

GPFPUD is shown in Figure 8-94 and described in Table 8-104.

Return to the Summary Table.

GPIO F Pull Up Disable Register (GPIO160 to 168)
Disables the Pull-Up on GPIO.
0: Enables the Pull-Up.
1: Disables the Pull-Up.
Reading the register returns the current value of the register setting.

**Figure 8-94. GPFPUD Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED |
| 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 |
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED |
| 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 | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | GPIO168 |
| 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 |
| GPIO167 | GPIO166 | GPIO165 | GPIO164 | GPIO163 | GPIO162 | GPIO161 | GPIO160 |
| R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h |

**Table 8-104. GPFPUD 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 |
| 11 | RESERVED | R/W | 1h | Reserved |
| 10 | RESERVED | R/W | 1h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-104. GPFPUD Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 9 | RESERVED | R/W | 1h | Reserved |
| 8 | GPIO168 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 7 | GPIO167 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 6 | GPIO166 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 5 | GPIO165 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 4 | GPIO164 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 3 | GPIO163 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 2 | GPIO162 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 1 | GPIO161 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |
| 0 | GPIO160 | R/W | 1h | Pull-Up Disable control for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.92 GPFINV Register (Offset = 150h) [Reset = 00000000h]

GPFINV is shown in Figure 8-95 and described in Table 8-105.

Return to the Summary Table.

GPIO F Input Polarity Invert Registers (GPIO160 to 168)
Selects between non-inverted and inverted GPIO input to the device.
0: selects non-inverted GPIO input
1: selects inverted GPIO input
Reading the register returns the current value of the register setting.

**Figure 8-95. GPFINV Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | 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 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | 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 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | GPIO168 |
| 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 |
| GPIO167 | GPIO166 | GPIO165 | GPIO164 | GPIO163 | GPIO162 | GPIO161 | GPIO160 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 8-105. GPFINV 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 |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-105. GPFINV Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 9 | RESERVED | R/W | 0h | Reserved |
| 8 | GPIO168 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 7 | GPIO167 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 6 | GPIO166 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 5 | GPIO165 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 4 | GPIO164 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 3 | GPIO163 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 2 | GPIO162 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 1 | GPIO161 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |
| 0 | GPIO160 | R/W | 0h | Input inversion control for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.93 GPFODR Register (Offset = 152h) [Reset = 00000000h]

GPFODR is shown in Figure 8-96 and described in Table 8-106.

Return to the Summary Table.

GPIO F Open Drain Output Register (GPIO160 to GPIO168)
Selects between normal and open-drain output for the GPIO pin.
0: Normal Output
1: Open Drain Output
Reading the register returns the current value of the register setting.
Note:
[1] In the Open Drain output mode, if the buffer is configured for output mode, a 0 value to be driven out comes out on the on the PAD while a 1 value to be driven out tri-states the buffer.

**Figure 8-96. GPFODR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | 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 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | 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 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | GPIO168 |
| 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 |
| GPIO167 | GPIO166 | GPIO165 | GPIO164 | GPIO163 | GPIO162 | GPIO161 | GPIO160 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 8-106. GPFODR 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 |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-106. GPFODR 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 | RESERVED | R/W | 0h | Reserved |
| 8 | GPIO168 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 7 | GPIO167 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 6 | GPIO166 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 5 | GPIO165 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 4 | GPIO164 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 3 | GPIO163 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 2 | GPIO162 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 1 | GPIO161 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |
| 0 | GPIO160 | R/W | 0h | Output Open-Drain control for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.94 GPFGMUX1 Register (Offset = 160h) [Reset = 00000000h]

GPFGMUX1 is shown in Figure 8-97 and described in Table 8-107.

Return to the Summary Table.

GPIO F Peripheral Group Mux (GPIO160 to 168)
Defines pin-muxing selection for GPIO.
Note: For complete pin-mux selection on GPIOx, GPAMUXy.GPIOx configuration is also required.

**Figure 8-97. GPFGMUX1 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 | | GPIO168 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO167 | | GPIO166 | | GPIO165 | | GPIO164 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO163 | | GPIO162 | | GPIO161 | | GPIO160 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-107. GPFGMUX1 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 | GPIO168 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 15-14 | GPIO167 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 13-12 | GPIO166 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 11-10 | GPIO165 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 9-8 | GPIO164 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 7-6 | GPIO163 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 5-4 | GPIO162 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |
| 3-2 | GPIO161 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-107. GPFGMUX1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1-0 | GPIO160 | R/W | 0h | Defines pin-muxing selection for GPIO<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.95 GPFCSEL1 Register (Offset = 168h) [Reset = 00000000h]

GPFCSEL1 is shown in Figure 8-98 and described in Table 8-108.

Return to the Summary Table.

GPIO F Core Select Register (GPIO160 to 167)
Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
xx00: CPU1 selected
xx01: CPU1.CLA1 selected
xx10: CPU2 selected
xx11: CPU2.CLA1 selected

**Figure 8-98. GPFCSEL1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO167 | | | | GPIO166 | | | | GPIO165 | | | | GPIO164 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO163 | | | | GPIO162 | | | | GPIO161 | | | | GPIO160 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |

**Table 8-108. GPFCSEL1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-28 | GPIO167 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 27-24 | GPIO166 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 23-20 | GPIO165 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 19-16 | GPIO164 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 15-12 | GPIO163 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 11-8 | GPIO162 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 7-4 | GPIO161 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |
| 3-0 | GPIO160 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.96 GPFCSEL2 Register (Offset = 16Ah) [Reset = 00000000h]

GPFCSEL2 is shown in Figure 8-99 and described in Table 8-109.

Return to the Summary Table.

GPIO F Core Select Register (GPIO168)
Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
xx00: CPU1 selected
xx01: CPU1.CLA1 selected
xx10: CPU2 selected
xx11: CPU2.CLA1 selected

**Figure 8-99. GPFCSEL2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 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 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | RESERVED | | | | RESERVED | | | | GPIO168 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |

**Table 8-109. GPFCSEL2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-28 | RESERVED | R/W | 0h | Reserved |
| 27-24 | RESERVED | R/W | 0h | Reserved |
| 23-20 | RESERVED | R/W | 0h | Reserved |
| 19-16 | RESERVED | R/W | 0h | Reserved |
| 15-12 | RESERVED | R/W | 0h | Reserved |
| 11-8 | RESERVED | R/W | 0h | Reserved |
| 7-4 | RESERVED | R/W | 0h | Reserved |
| 3-0 | GPIO168 | R/W | 0h | Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.97 GPFLOCK Register (Offset = 17Ch) [Reset = 00000000h]

GPFLOCK is shown in Figure 8-100 and described in Table 8-110.

Return to the Summary Table.

GPIO F Lock Configuration Register (GPIO160 to 168)
GPIO Configuration Lock for GPIO.
0: Bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx register which control the same pin can be changed
1: Locks changes to the bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx registers which control the same pin

**Figure 8-100. GPFLOCK Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | 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 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | 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 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | GPIO168 |
| 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 |
| GPIO167 | GPIO166 | GPIO165 | GPIO164 | GPIO163 | GPIO162 | GPIO161 | GPIO160 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 8-110. GPFLOCK 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 |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-110. GPFLOCK Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 10 | RESERVED | R/W | 0h | Reserved |
| 9 | RESERVED | R/W | 0h | Reserved |
| 8 | GPIO168 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 7 | GPIO167 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 6 | GPIO166 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 5 | GPIO165 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 4 | GPIO164 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 3 | GPIO163 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 2 | GPIO162 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 1 | GPIO161 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |
| 0 | GPIO160 | R/W | 0h | Configuration Lock bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.2.98 GPFCR Register (Offset = 17Eh) [Reset = 00000000h]

GPFCR is shown in Figure 8-101 and described in Table 8-111.

Return to the Summary Table.

GPIO F Lock Commit Register (GPIO160 to 168)
GPIO Configuration Lock Commit for GPIO:
1: Locks changes to the bit in GPyLOCK register which controls the same pin
0: Bit in the GPyLOCK register which controls the same pin can be changed

**Figure 8-101. GPFCR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | 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 |
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | 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 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | GPIO168 |
| 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 |
| GPIO167 | GPIO166 | GPIO165 | GPIO164 | GPIO163 | GPIO162 | GPIO161 | GPIO160 |
| R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h |

**Table 8-111. GPFCR 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 |
| 9 | RESERVED | R/WOnce | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-111. GPFCR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 8 | GPIO168 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 7 | GPIO167 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 6 | GPIO166 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 5 | GPIO165 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 4 | GPIO164 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 3 | GPIO163 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 2 | GPIO162 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 1 | GPIO161 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |
| 0 | GPIO160 | R/WOnce | 0h | Configuration lock commit bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.3 GPIO_DATA_REGS Registers

Table 8-112 lists the memory-mapped registers for the GPIO_DATA_REGS registers. All register offset addresses not listed in Table 8-112 should be considered as reserved locations and the register contents should not be modified.

**Table 8-112. GPIO_DATA_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | GPADAT | GPIO A Data Register (GPIO0 to 31) | | Go |
| 2h | GPASET | GPIO A Data Set Register (GPIO0 to 31) | | Go |
| 4h | GPACLEAR | GPIO A Data Clear Register (GPIO0 to 31) | | Go |
| 6h | GPATOGGLE | GPIO A Data Toggle Register (GPIO0 to 31) | | Go |
| 8h | GPBDAT | GPIO B Data Register (GPIO32 to 63) | | Go |
| Ah | GPBSET | GPIO B Data Set Register (GPIO32 to 63) | | Go |
| Ch | GPBCLEAR | GPIO B Data Clear Register (GPIO32 to 63) | | Go |
| Eh | GPBTOGGLE | GPIO B Data Toggle Register (GPIO32 to 63) | | Go |
| 10h | GPCDAT | GPIO C Data Register (GPIO64 to 95) | | Go |
| 12h | GPCSET | GPIO C Data Set Register (GPIO64 to 95) | | Go |
| 14h | GPCCLEAR | GPIO C Data Clear Register (GPIO64 to 95) | | Go |
| 16h | GPCTOGGLE | GPIO C Data Toggle Register (GPIO64 to 95) | | Go |
| 18h | GPDDAT | GPIO D Data Register (GPIO96 to 127) | | Go |
| 1Ah | GPDSET | GPIO D Data Set Register (GPIO96 to 127) | | Go |
| 1Ch | GPDCLEAR | GPIO D Data Clear Register (GPIO96 to 127) | | Go |
| 1Eh | GPDTOGGLE | GPIO D Data Toggle Register (GPIO96 to 127) | | Go |
| 20h | GPEDAT | GPIO E Data Register (GPIO128 to 159) | | Go |
| 22h | GPESET | GPIO E Data Set Register (GPIO128 to 159) | | Go |
| 24h | GPECLEAR | GPIO E Data Clear Register (GPIO128 to 159) | | Go |
| 26h | GPETOGGLE | GPIO E Data Toggle Register (GPIO128 to 159) | | Go |
| 28h | GPFDAT | GPIO F Data Register (GPIO160 to 168) | | Go |
| 2Ah | GPFSET | GPIO F Data Set Register (GPIO160 to 168) | | Go |
| 2Ch | GPFCLEAR | GPIO F Data Clear Register (GPIO160 to 168) | | Go |
| 2Eh | GPFTOGGLE | GPIO F Data Toggle Register (GPIO160 to 168) | | Go |

Complex bit access types are encoded to fit into small table cells. Table 8-113 shows the codes that are used for access types in this section.

**Table 8-113. GPIO_DATA_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. |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-113. GPIO_DATA_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. |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.3.1 GPADAT Register (Offset = 0h) [Reset = 00000000h]

GPADAT is shown in Figure 8-102 and described in Table 8-114.

Return to the Summary Table.

GPIO A Data Register (GPIO0 to 31)
Reading this register reflects the current state of the GPIO pin regardless of which mode the GPIO is in.
Writing to this register will set the GPIO pin high or low if the pin is enabled for GPIO output mode. If the GPIO is not in output mode the value written is latched but will not be reflected on the GPIO pin or reads of the GPxDAT register. The written value latched will become active when the GPIO is put into GPIO Output mode
A system reset will clear all bits and latched values to zero.
NOTE: Bit-wise read-modify-write operations should not be performed on this register. For bit-wise operations the GPxSET, GPxCLEAR, or GPxTOGGLE registers should be used instead. If direct writes to GPxDAT are necessary, the entire register should be written at one time.

**Figure 8-102. GPADAT Register**

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

**Table 8-114. GPADAT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO31 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 30 | GPIO30 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 29 | GPIO29 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 28 | GPIO28 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 27 | GPIO27 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 26 | GPIO26 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 25 | GPIO25 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 24 | GPIO24 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 23 | GPIO23 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 22 | GPIO22 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-114. GPADAT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 21 | GPIO21 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 20 | GPIO20 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 19 | GPIO19 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 18 | GPIO18 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 17 | GPIO17 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 16 | GPIO16 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 15 | GPIO15 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 14 | GPIO14 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 13 | GPIO13 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 12 | GPIO12 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 11 | GPIO11 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 10 | GPIO10 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 9 | GPIO9 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 8 | GPIO8 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 7 | GPIO7 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 6 | GPIO6 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 5 | GPIO5 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 4 | GPIO4 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 3 | GPIO3 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 2 | GPIO2 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 1 | GPIO1 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 0 | GPIO0 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.3.2 GPASET Register (Offset = 2h) [Reset = 00000000h]

GPASET is shown in Figure 8-103 and described in Table 8-115.

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 8-103. GPASET Register**

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

**Table 8-115. GPASET Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO31 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 30 | GPIO30 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 29 | GPIO29 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 28 | GPIO28 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 27 | GPIO27 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 26 | GPIO26 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 25 | GPIO25 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 24 | GPIO24 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 23 | GPIO23 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 22 | GPIO22 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 21 | GPIO21 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 20 | GPIO20 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-115. GPASET Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19 | GPIO19 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 18 | GPIO18 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 17 | GPIO17 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 16 | GPIO16 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 15 | GPIO15 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 14 | GPIO14 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 13 | GPIO13 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 12 | GPIO12 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 11 | GPIO11 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 10 | GPIO10 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 9 | GPIO9 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 8 | GPIO8 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 7 | GPIO7 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 6 | GPIO6 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 5 | GPIO5 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 4 | GPIO4 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 3 | GPIO3 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 2 | GPIO2 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 1 | GPIO1 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 0 | GPIO0 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.3.3 GPACLEAR Register (Offset = 4h) [Reset = 00000000h]

GPACLEAR is shown in Figure 8-104 and described in Table 8-116.

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 8-104. GPACLEAR Register**

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

**Table 8-116. GPACLEAR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO31 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 30 | GPIO30 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 29 | GPIO29 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 28 | GPIO28 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 27 | GPIO27 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 26 | GPIO26 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 25 | GPIO25 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 24 | GPIO24 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 23 | GPIO23 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 22 | GPIO22 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 21 | GPIO21 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 20 | GPIO20 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-116. GPACLEAR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19 | GPIO19 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 18 | GPIO18 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 17 | GPIO17 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 16 | GPIO16 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 15 | GPIO15 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 14 | GPIO14 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 13 | GPIO13 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 12 | GPIO12 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 11 | GPIO11 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 10 | GPIO10 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 9 | GPIO9 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 8 | GPIO8 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 7 | GPIO7 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 6 | GPIO6 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 5 | GPIO5 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 4 | GPIO4 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 3 | GPIO3 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 2 | GPIO2 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 1 | GPIO1 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 0 | GPIO0 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.3.4 GPATOGGLE Register (Offset = 6h) [Reset = 00000000h]

GPATOGGLE is shown in Figure 8-105 and described in Table 8-117.

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 8-105. GPATOGGLE Register**

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

**Table 8-117. GPATOGGLE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO31 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 30 | GPIO30 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 29 | GPIO29 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 28 | GPIO28 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 27 | GPIO27 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 26 | GPIO26 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 25 | GPIO25 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 24 | GPIO24 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 23 | GPIO23 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 22 | GPIO22 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 21 | GPIO21 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 20 | GPIO20 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-117. GPATOGGLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19 | GPIO19 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 18 | GPIO18 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 17 | GPIO17 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 16 | GPIO16 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 15 | GPIO15 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 14 | GPIO14 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 13 | GPIO13 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 12 | GPIO12 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 11 | GPIO11 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 10 | GPIO10 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 9 | GPIO9 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 8 | GPIO8 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 7 | GPIO7 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 6 | GPIO6 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 5 | GPIO5 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 4 | GPIO4 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 3 | GPIO3 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 2 | GPIO2 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 1 | GPIO1 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 0 | GPIO0 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.3.5 GPBDAT Register (Offset = 8h) [Reset = 00000000h]

GPBDAT is shown in Figure 8-106 and described in Table 8-118.

Return to the Summary Table.

GPIO B Data Register (GPIO32 to 63)
Reading this register reflects the current state of the GPIO pin regardless of which mode the GPIO is in.
Writing to this register will set the GPIO pin high or low if the pin is enabled for GPIO output mode. If the GPIO is not in output mode the value written is latched but will not be reflected on the GPIO pin or reads of the GPxDAT register. The written value latched will become active when the GPIO is put into GPIO Output mode
A system reset will clear all bits and latched values to zero.
NOTE: Bit-wise read-modify-write operations should not be performed on this register. For bit-wise operations the GPxSET, GPxCLEAR, or GPxTOGGLE registers should be used instead. If direct writes to GPxDAT are necessary, the entire register should be written at one time.

**Figure 8-106. GPBDAT Register**

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

**Table 8-118. GPBDAT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO63 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 30 | GPIO62 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 29 | GPIO61 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 28 | GPIO60 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 27 | GPIO59 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 26 | GPIO58 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 25 | GPIO57 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 24 | GPIO56 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 23 | GPIO55 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 22 | GPIO54 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-118. GPBDAT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 21 | GPIO53 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 20 | GPIO52 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 19 | GPIO51 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 18 | GPIO50 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 17 | GPIO49 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 16 | GPIO48 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 15 | GPIO47 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 14 | GPIO46 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 13 | GPIO45 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 12 | GPIO44 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 11 | GPIO43 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 10 | GPIO42 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 9 | GPIO41 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 8 | GPIO40 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 7 | GPIO39 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 6 | GPIO38 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 5 | GPIO37 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 4 | GPIO36 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 3 | GPIO35 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 2 | GPIO34 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 1 | GPIO33 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 0 | GPIO32 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.3.6 GPBSET Register (Offset = Ah) [Reset = 00000000h]

GPBSET is shown in Figure 8-107 and described in Table 8-119.

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 8-107. GPBSET Register**

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

**Table 8-119. GPBSET Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO63 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 30 | GPIO62 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 29 | GPIO61 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 28 | GPIO60 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 27 | GPIO59 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 26 | GPIO58 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 25 | GPIO57 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 24 | GPIO56 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 23 | GPIO55 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 22 | GPIO54 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 21 | GPIO53 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 20 | GPIO52 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-119. GPBSET Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19 | GPIO51 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 18 | GPIO50 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 17 | GPIO49 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 16 | GPIO48 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 15 | GPIO47 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 14 | GPIO46 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 13 | GPIO45 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 12 | GPIO44 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 11 | GPIO43 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 10 | GPIO42 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 9 | GPIO41 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 8 | GPIO40 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 7 | GPIO39 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 6 | GPIO38 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 5 | GPIO37 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 4 | GPIO36 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 3 | GPIO35 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 2 | GPIO34 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 1 | GPIO33 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 0 | GPIO32 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.3.7 GPBCLEAR Register (Offset = Ch) [Reset = 00000000h]

GPBCLEAR is shown in Figure 8-108 and described in Table 8-120.

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 8-108. GPBCLEAR Register**

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

**Table 8-120. GPBCLEAR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO63 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 30 | GPIO62 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 29 | GPIO61 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 28 | GPIO60 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 27 | GPIO59 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 26 | GPIO58 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 25 | GPIO57 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 24 | GPIO56 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 23 | GPIO55 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 22 | GPIO54 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 21 | GPIO53 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 20 | GPIO52 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-120. GPBCLEAR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19 | GPIO51 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 18 | GPIO50 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 17 | GPIO49 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 16 | GPIO48 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 15 | GPIO47 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 14 | GPIO46 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 13 | GPIO45 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 12 | GPIO44 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 11 | GPIO43 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 10 | GPIO42 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 9 | GPIO41 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 8 | GPIO40 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 7 | GPIO39 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 6 | GPIO38 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 5 | GPIO37 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 4 | GPIO36 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 3 | GPIO35 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 2 | GPIO34 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 1 | GPIO33 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 0 | GPIO32 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.3.8 GPBTOGGLE Register (Offset = Eh) [Reset = 00000000h]

GPBTOGGLE is shown in Figure 8-109 and described in Table 8-121.

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 8-109. GPBTOGGLE Register**

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

**Table 8-121. GPBTOGGLE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO63 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 30 | GPIO62 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 29 | GPIO61 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 28 | GPIO60 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 27 | GPIO59 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 26 | GPIO58 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 25 | GPIO57 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 24 | GPIO56 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 23 | GPIO55 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 22 | GPIO54 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 21 | GPIO53 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 20 | GPIO52 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-121. GPBTOGGLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19 | GPIO51 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 18 | GPIO50 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 17 | GPIO49 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 16 | GPIO48 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 15 | GPIO47 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 14 | GPIO46 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 13 | GPIO45 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 12 | GPIO44 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 11 | GPIO43 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 10 | GPIO42 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 9 | GPIO41 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 8 | GPIO40 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 7 | GPIO39 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 6 | GPIO38 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 5 | GPIO37 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 4 | GPIO36 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 3 | GPIO35 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 2 | GPIO34 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 1 | GPIO33 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 0 | GPIO32 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.3.9 GPCDAT Register (Offset = 10h) [Reset = 00000000h]

GPCDAT is shown in Figure 8-110 and described in Table 8-122.

Return to the Summary Table.

GPIO C Data Register (GPIO64 to 95)
Reading this register reflects the current state of the GPIO pin regardless of which mode the GPIO is in.
Writing to this register will set the GPIO pin high or low if the pin is enabled for GPIO output mode. If the GPIO is not in output mode the value written is latched but will not be reflected on the GPIO pin or reads of the GPxDAT register. The written value latched will become active when the GPIO is put into GPIO Output mode
A system reset will clear all bits and latched values to zero.
NOTE: Bit-wise read-modify-write operations should not be performed on this register. For bit-wise operations the GPxSET, GPxCLEAR, or GPxTOGGLE registers should be used instead. If direct writes to GPxDAT are necessary, the entire register should be written at one time.

**Figure 8-110. GPCDAT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO95 | GPIO94 | GPIO93 | GPIO92 | GPIO91 | GPIO90 | GPIO89 | GPIO88 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO87 | GPIO86 | GPIO85 | GPIO84 | GPIO83 | GPIO82 | GPIO81 | GPIO80 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO79 | GPIO78 | GPIO77 | GPIO76 | GPIO75 | GPIO74 | GPIO73 | GPIO72 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO71 | GPIO70 | GPIO69 | GPIO68 | GPIO67 | GPIO66 | GPIO65 | GPIO64 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 8-122. GPCDAT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO95 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 30 | GPIO94 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 29 | GPIO93 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 28 | GPIO92 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 27 | GPIO91 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 26 | GPIO90 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 25 | GPIO89 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 24 | GPIO88 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 23 | GPIO87 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 22 | GPIO86 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-122. GPCDAT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 21 | GPIO85 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 20 | GPIO84 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 19 | GPIO83 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 18 | GPIO82 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 17 | GPIO81 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 16 | GPIO80 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 15 | GPIO79 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 14 | GPIO78 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 13 | GPIO77 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 12 | GPIO76 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 11 | GPIO75 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 10 | GPIO74 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 9 | GPIO73 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 8 | GPIO72 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 7 | GPIO71 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 6 | GPIO70 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 5 | GPIO69 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 4 | GPIO68 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 3 | GPIO67 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 2 | GPIO66 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 1 | GPIO65 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 0 | GPIO64 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.3.10 GPCSET Register (Offset = 12h) [Reset = 00000000h]

GPCSET is shown in Figure 8-111 and described in Table 8-123.

Return to the Summary Table.

GPIO C Data Set Register (GPIO64 to 95)
Writing a 1 will force GPIO output data latch to 1.
Writes of 0 are ignored.
Always reads back a 0.

**Figure 8-111. GPCSET Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO95 | GPIO94 | GPIO93 | GPIO92 | GPIO91 | GPIO90 | GPIO89 | GPIO88 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO87 | GPIO86 | GPIO85 | GPIO84 | GPIO83 | GPIO82 | GPIO81 | GPIO80 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO79 | GPIO78 | GPIO77 | GPIO76 | GPIO75 | GPIO74 | GPIO73 | GPIO72 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO71 | GPIO70 | GPIO69 | GPIO68 | GPIO67 | GPIO66 | GPIO65 | GPIO64 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 8-123. GPCSET Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO95 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 30 | GPIO94 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 29 | GPIO93 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 28 | GPIO92 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 27 | GPIO91 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 26 | GPIO90 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 25 | GPIO89 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 24 | GPIO88 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 23 | GPIO87 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 22 | GPIO86 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 21 | GPIO85 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 20 | GPIO84 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-123. GPCSET Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19 | GPIO83 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 18 | GPIO82 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 17 | GPIO81 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 16 | GPIO80 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 15 | GPIO79 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 14 | GPIO78 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 13 | GPIO77 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 12 | GPIO76 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 11 | GPIO75 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 10 | GPIO74 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 9 | GPIO73 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 8 | GPIO72 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 7 | GPIO71 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 6 | GPIO70 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 5 | GPIO69 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 4 | GPIO68 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 3 | GPIO67 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 2 | GPIO66 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 1 | GPIO65 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 0 | GPIO64 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.3.11 GPCCLEAR Register (Offset = 14h) [Reset = 00000000h]

GPCCLEAR is shown in Figure 8-112 and described in Table 8-124.

Return to the Summary Table.

GPIO C Data Clear Register (GPIO64 to 95)
Writing a 1 will force GPIO0 output data latch to 0.
Writes of 0 are ignored.
Always reads back a 0.

**Figure 8-112. GPCCLEAR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO95 | GPIO94 | GPIO93 | GPIO92 | GPIO91 | GPIO90 | GPIO89 | GPIO88 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO87 | GPIO86 | GPIO85 | GPIO84 | GPIO83 | GPIO82 | GPIO81 | GPIO80 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO79 | GPIO78 | GPIO77 | GPIO76 | GPIO75 | GPIO74 | GPIO73 | GPIO72 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO71 | GPIO70 | GPIO69 | GPIO68 | GPIO67 | GPIO66 | GPIO65 | GPIO64 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 8-124. GPCCLEAR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO95 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 30 | GPIO94 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 29 | GPIO93 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 28 | GPIO92 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 27 | GPIO91 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 26 | GPIO90 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 25 | GPIO89 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 24 | GPIO88 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 23 | GPIO87 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 22 | GPIO86 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 21 | GPIO85 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 20 | GPIO84 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-124. GPCCLEAR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19 | GPIO83 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 18 | GPIO82 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 17 | GPIO81 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 16 | GPIO80 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 15 | GPIO79 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 14 | GPIO78 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 13 | GPIO77 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 12 | GPIO76 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 11 | GPIO75 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 10 | GPIO74 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 9 | GPIO73 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 8 | GPIO72 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 7 | GPIO71 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 6 | GPIO70 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 5 | GPIO69 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 4 | GPIO68 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 3 | GPIO67 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 2 | GPIO66 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 1 | GPIO65 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 0 | GPIO64 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.3.12 GPCTOGGLE Register (Offset = 16h) [Reset = 00000000h]

GPCTOGGLE is shown in Figure 8-113 and described in Table 8-125.

Return to the Summary Table.

GPIO C Data Toggle Register (GPIO64 to 95)
Writing a 1 will toggle GPIO0 output data latch 1 to 0 or 0 to 1.
Writes of 0 are ignored.
Always reads back a 0.

**Figure 8-113. GPCTOGGLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO95 | GPIO94 | GPIO93 | GPIO92 | GPIO91 | GPIO90 | GPIO89 | GPIO88 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO87 | GPIO86 | GPIO85 | GPIO84 | GPIO83 | GPIO82 | GPIO81 | GPIO80 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO79 | GPIO78 | GPIO77 | GPIO76 | GPIO75 | GPIO74 | GPIO73 | GPIO72 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO71 | GPIO70 | GPIO69 | GPIO68 | GPIO67 | GPIO66 | GPIO65 | GPIO64 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 8-125. GPCTOGGLE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO95 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 30 | GPIO94 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 29 | GPIO93 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 28 | GPIO92 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 27 | GPIO91 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 26 | GPIO90 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 25 | GPIO89 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 24 | GPIO88 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 23 | GPIO87 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 22 | GPIO86 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 21 | GPIO85 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 20 | GPIO84 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-125. GPCTOGGLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19 | GPIO83 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 18 | GPIO82 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 17 | GPIO81 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 16 | GPIO80 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 15 | GPIO79 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 14 | GPIO78 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 13 | GPIO77 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 12 | GPIO76 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 11 | GPIO75 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 10 | GPIO74 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 9 | GPIO73 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 8 | GPIO72 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 7 | GPIO71 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 6 | GPIO70 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 5 | GPIO69 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 4 | GPIO68 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 3 | GPIO67 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 2 | GPIO66 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 1 | GPIO65 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 0 | GPIO64 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.3.13 GPDDAT Register (Offset = 18h) [Reset = 00000000h]

GPDDAT is shown in Figure 8-114 and described in Table 8-126.

Return to the Summary Table.

GPIO D Data Register (GPIO96 to 127)
Reading this register reflects the current state of the GPIO pin regardless of which mode the GPIO is in.
Writing to this register will set the GPIO pin high or low if the pin is enabled for GPIO output mode. If the GPIO is not in output mode the value written is latched but will not be reflected on the GPIO pin or reads of the GPxDAT register. The written value latched will become active when the GPIO is put into GPIO Output mode
A system reset will clear all bits and latched values to zero.
NOTE: Bit-wise read-modify-write operations should not be performed on this register. For bit-wise operations the GPxSET, GPxCLEAR, or GPxTOGGLE registers should be used instead. If direct writes to GPxDAT are necessary, the entire register should be written at one time.

**Figure 8-114. GPDDAT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO127 | GPIO126 | GPIO125 | GPIO124 | GPIO123 | GPIO122 | GPIO121 | GPIO120 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO119 | GPIO118 | GPIO117 | GPIO116 | GPIO115 | GPIO114 | GPIO113 | GPIO112 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO111 | GPIO110 | GPIO109 | GPIO108 | GPIO107 | GPIO106 | GPIO105 | GPIO104 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO103 | GPIO102 | GPIO101 | GPIO100 | GPIO99 | GPIO98 | GPIO97 | GPIO96 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 8-126. GPDDAT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO127 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 30 | GPIO126 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 29 | GPIO125 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 28 | GPIO124 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 27 | GPIO123 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 26 | GPIO122 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 25 | GPIO121 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 24 | GPIO120 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 23 | GPIO119 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 22 | GPIO118 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-126. GPDDAT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 21 | GPIO117 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 20 | GPIO116 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 19 | GPIO115 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 18 | GPIO114 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 17 | GPIO113 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 16 | GPIO112 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 15 | GPIO111 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 14 | GPIO110 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 13 | GPIO109 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 12 | GPIO108 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 11 | GPIO107 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 10 | GPIO106 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 9 | GPIO105 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 8 | GPIO104 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 7 | GPIO103 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 6 | GPIO102 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 5 | GPIO101 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 4 | GPIO100 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 3 | GPIO99 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 2 | GPIO98 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 1 | GPIO97 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 0 | GPIO96 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.3.14 GPDSET Register (Offset = 1Ah) [Reset = 00000000h]

GPDSET is shown in Figure 8-115 and described in Table 8-127.

Return to the Summary Table.

GPIO D Data Set Register (GPIO96 to 127)
Writing a 1 will force GPIO output data latch to 1.
Writes of 0 are ignored.
Always reads back a 0.

**Figure 8-115. GPDSET Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO127 | GPIO126 | GPIO125 | GPIO124 | GPIO123 | GPIO122 | GPIO121 | GPIO120 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO119 | GPIO118 | GPIO117 | GPIO116 | GPIO115 | GPIO114 | GPIO113 | GPIO112 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO111 | GPIO110 | GPIO109 | GPIO108 | GPIO107 | GPIO106 | GPIO105 | GPIO104 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO103 | GPIO102 | GPIO101 | GPIO100 | GPIO99 | GPIO98 | GPIO97 | GPIO96 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 8-127. GPDSET Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO127 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 30 | GPIO126 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 29 | GPIO125 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 28 | GPIO124 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 27 | GPIO123 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 26 | GPIO122 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 25 | GPIO121 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 24 | GPIO120 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 23 | GPIO119 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 22 | GPIO118 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 21 | GPIO117 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 20 | GPIO116 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-127. GPDSET Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19 | GPIO115 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 18 | GPIO114 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 17 | GPIO113 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 16 | GPIO112 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 15 | GPIO111 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 14 | GPIO110 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 13 | GPIO109 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 12 | GPIO108 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 11 | GPIO107 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 10 | GPIO106 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 9 | GPIO105 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 8 | GPIO104 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 7 | GPIO103 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 6 | GPIO102 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 5 | GPIO101 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 4 | GPIO100 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 3 | GPIO99 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 2 | GPIO98 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 1 | GPIO97 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 0 | GPIO96 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.3.15 GPDCLEAR Register (Offset = 1Ch) [Reset = 00000000h]

GPDCLEAR is shown in Figure 8-116 and described in Table 8-128.

Return to the Summary Table.

GPIO D Data Clear Register (GPIO96 to 127)
Writing a 1 will force GPIO0 output data latch to 0.
Writes of 0 are ignored.
Always reads back a 0.

**Figure 8-116. GPDCLEAR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO127 | GPIO126 | GPIO125 | GPIO124 | GPIO123 | GPIO122 | GPIO121 | GPIO120 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO119 | GPIO118 | GPIO117 | GPIO116 | GPIO115 | GPIO114 | GPIO113 | GPIO112 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO111 | GPIO110 | GPIO109 | GPIO108 | GPIO107 | GPIO106 | GPIO105 | GPIO104 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO103 | GPIO102 | GPIO101 | GPIO100 | GPIO99 | GPIO98 | GPIO97 | GPIO96 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 8-128. GPDCLEAR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO127 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 30 | GPIO126 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 29 | GPIO125 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 28 | GPIO124 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 27 | GPIO123 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 26 | GPIO122 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 25 | GPIO121 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 24 | GPIO120 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 23 | GPIO119 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 22 | GPIO118 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 21 | GPIO117 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 20 | GPIO116 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-128. GPDCLEAR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19 | GPIO115 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 18 | GPIO114 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 17 | GPIO113 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 16 | GPIO112 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 15 | GPIO111 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 14 | GPIO110 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 13 | GPIO109 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 12 | GPIO108 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 11 | GPIO107 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 10 | GPIO106 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 9 | GPIO105 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 8 | GPIO104 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 7 | GPIO103 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 6 | GPIO102 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 5 | GPIO101 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 4 | GPIO100 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 3 | GPIO99 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 2 | GPIO98 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 1 | GPIO97 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 0 | GPIO96 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.3.16 GPDTOGGLE Register (Offset = 1Eh) [Reset = 00000000h]

GPDTOGGLE is shown in Figure 8-117 and described in Table 8-129.

Return to the Summary Table.

GPIO D Data Toggle Register (GPIO96 to 127)
Writing a 1 will toggle GPIO0 output data latch 1 to 0 or 0 to 1.
Writes of 0 are ignored.
Always reads back a 0.

**Figure 8-117. GPDTOGGLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO127 | GPIO126 | GPIO125 | GPIO124 | GPIO123 | GPIO122 | GPIO121 | GPIO120 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO119 | GPIO118 | GPIO117 | GPIO116 | GPIO115 | GPIO114 | GPIO113 | GPIO112 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO111 | GPIO110 | GPIO109 | GPIO108 | GPIO107 | GPIO106 | GPIO105 | GPIO104 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO103 | GPIO102 | GPIO101 | GPIO100 | GPIO99 | GPIO98 | GPIO97 | GPIO96 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 8-129. GPDTOGGLE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO127 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 30 | GPIO126 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 29 | GPIO125 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 28 | GPIO124 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 27 | GPIO123 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 26 | GPIO122 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 25 | GPIO121 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 24 | GPIO120 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 23 | GPIO119 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 22 | GPIO118 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 21 | GPIO117 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 20 | GPIO116 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-129. GPDTOGGLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19 | GPIO115 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 18 | GPIO114 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 17 | GPIO113 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 16 | GPIO112 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 15 | GPIO111 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 14 | GPIO110 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 13 | GPIO109 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 12 | GPIO108 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 11 | GPIO107 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 10 | GPIO106 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 9 | GPIO105 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 8 | GPIO104 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 7 | GPIO103 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 6 | GPIO102 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 5 | GPIO101 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 4 | GPIO100 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 3 | GPIO99 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 2 | GPIO98 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 1 | GPIO97 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 0 | GPIO96 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.3.17 GPEDAT Register (Offset = 20h) [Reset = 00000000h]

GPEDAT is shown in Figure 8-118 and described in Table 8-130.

Return to the Summary Table.

GPIO E Data Register (GPIO128 to 159)
Reading this register reflects the current state of the GPIO pin regardless of which mode the GPIO is in.
Writing to this register will set the GPIO pin high or low if the pin is enabled for GPIO output mode. If the GPIO is not in output mode the value written is latched but will not be reflected on the GPIO pin or reads of the GPxDAT register. The written value latched will become active when the GPIO is put into GPIO Output mode
A system reset will clear all bits and latched values to zero.
NOTE: Bit-wise read-modify-write operations should not be performed on this register. For bit-wise operations the GPxSET, GPxCLEAR, or GPxTOGGLE registers should be used instead. If direct writes to GPxDAT are necessary, the entire register should be written at one time.

**Figure 8-118. GPEDAT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO159 | GPIO158 | GPIO157 | GPIO156 | GPIO155 | GPIO154 | GPIO153 | GPIO152 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO151 | GPIO150 | GPIO149 | GPIO148 | GPIO147 | GPIO146 | GPIO145 | GPIO144 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO143 | GPIO142 | GPIO141 | GPIO140 | GPIO139 | GPIO138 | GPIO137 | GPIO136 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO135 | GPIO134 | GPIO133 | GPIO132 | GPIO131 | GPIO130 | GPIO129 | GPIO128 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 8-130. GPEDAT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO159 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 30 | GPIO158 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 29 | GPIO157 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 28 | GPIO156 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 27 | GPIO155 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 26 | GPIO154 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 25 | GPIO153 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 24 | GPIO152 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 23 | GPIO151 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 22 | GPIO150 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-130. GPEDAT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 21 | GPIO149 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 20 | GPIO148 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 19 | GPIO147 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 18 | GPIO146 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 17 | GPIO145 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 16 | GPIO144 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 15 | GPIO143 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 14 | GPIO142 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 13 | GPIO141 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 12 | GPIO140 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 11 | GPIO139 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 10 | GPIO138 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 9 | GPIO137 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 8 | GPIO136 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 7 | GPIO135 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 6 | GPIO134 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 5 | GPIO133 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 4 | GPIO132 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 3 | GPIO131 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 2 | GPIO130 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 1 | GPIO129 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 0 | GPIO128 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.3.18 GPESET Register (Offset = 22h) [Reset = 00000000h]

GPESET is shown in Figure 8-119 and described in Table 8-131.

Return to the Summary Table.

GPIO E Data Set Register (GPIO128 to 159)
Writing a 1 will force GPIO output data latch to 1.
Writes of 0 are ignored.
Always reads back a 0.

**Figure 8-119. GPESET Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO159 | GPIO158 | GPIO157 | GPIO156 | GPIO155 | GPIO154 | GPIO153 | GPIO152 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO151 | GPIO150 | GPIO149 | GPIO148 | GPIO147 | GPIO146 | GPIO145 | GPIO144 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO143 | GPIO142 | GPIO141 | GPIO140 | GPIO139 | GPIO138 | GPIO137 | GPIO136 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO135 | GPIO134 | GPIO133 | GPIO132 | GPIO131 | GPIO130 | GPIO129 | GPIO128 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 8-131. GPESET Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO159 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 30 | GPIO158 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 29 | GPIO157 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 28 | GPIO156 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 27 | GPIO155 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 26 | GPIO154 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 25 | GPIO153 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 24 | GPIO152 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 23 | GPIO151 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 22 | GPIO150 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 21 | GPIO149 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 20 | GPIO148 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-131. GPESET Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19 | GPIO147 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 18 | GPIO146 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 17 | GPIO145 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 16 | GPIO144 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 15 | GPIO143 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 14 | GPIO142 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 13 | GPIO141 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 12 | GPIO140 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 11 | GPIO139 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 10 | GPIO138 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 9 | GPIO137 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 8 | GPIO136 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 7 | GPIO135 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 6 | GPIO134 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 5 | GPIO133 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 4 | GPIO132 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 3 | GPIO131 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 2 | GPIO130 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 1 | GPIO129 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 0 | GPIO128 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.3.19 GPECLEAR Register (Offset = 24h) [Reset = 00000000h]

GPECLEAR is shown in Figure 8-120 and described in Table 8-132.

Return to the Summary Table.

GPIO E Data Clear Register (GPIO128 to 159)
Writing a 1 will force GPIO0 output data latch to 0.
Writes of 0 are ignored.
Always reads back a 0.

**Figure 8-120. GPECLEAR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO159 | GPIO158 | GPIO157 | GPIO156 | GPIO155 | GPIO154 | GPIO153 | GPIO152 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO151 | GPIO150 | GPIO149 | GPIO148 | GPIO147 | GPIO146 | GPIO145 | GPIO144 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO143 | GPIO142 | GPIO141 | GPIO140 | GPIO139 | GPIO138 | GPIO137 | GPIO136 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO135 | GPIO134 | GPIO133 | GPIO132 | GPIO131 | GPIO130 | GPIO129 | GPIO128 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 8-132. GPECLEAR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO159 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 30 | GPIO158 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 29 | GPIO157 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 28 | GPIO156 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 27 | GPIO155 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 26 | GPIO154 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 25 | GPIO153 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 24 | GPIO152 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 23 | GPIO151 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 22 | GPIO150 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 21 | GPIO149 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 20 | GPIO148 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-132. GPECLEAR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19 | GPIO147 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 18 | GPIO146 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 17 | GPIO145 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 16 | GPIO144 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 15 | GPIO143 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 14 | GPIO142 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 13 | GPIO141 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 12 | GPIO140 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 11 | GPIO139 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 10 | GPIO138 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 9 | GPIO137 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 8 | GPIO136 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 7 | GPIO135 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 6 | GPIO134 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 5 | GPIO133 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 4 | GPIO132 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 3 | GPIO131 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 2 | GPIO130 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 1 | GPIO129 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 0 | GPIO128 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.3.20 GPETOGGLE Register (Offset = 26h) [Reset = 00000000h]

GPETOGGLE is shown in Figure 8-121 and described in Table 8-133.

Return to the Summary Table.

GPIO E Data Toggle Register (GPIO128 to 159)
Writing a 1 will toggle GPIO0 output data latch 1 to 0 or 0 to 1.
Writes of 0 are ignored.
Always reads back a 0.

**Figure 8-121. GPETOGGLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO159 | GPIO158 | GPIO157 | GPIO156 | GPIO155 | GPIO154 | GPIO153 | GPIO152 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO151 | GPIO150 | GPIO149 | GPIO148 | GPIO147 | GPIO146 | GPIO145 | GPIO144 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO143 | GPIO142 | GPIO141 | GPIO140 | GPIO139 | GPIO138 | GPIO137 | GPIO136 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO135 | GPIO134 | GPIO133 | GPIO132 | GPIO131 | GPIO130 | GPIO129 | GPIO128 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 8-133. GPETOGGLE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO159 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 30 | GPIO158 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 29 | GPIO157 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 28 | GPIO156 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 27 | GPIO155 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 26 | GPIO154 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 25 | GPIO153 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 24 | GPIO152 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 23 | GPIO151 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 22 | GPIO150 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 21 | GPIO149 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 20 | GPIO148 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-133. GPETOGGLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19 | GPIO147 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 18 | GPIO146 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 17 | GPIO145 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 16 | GPIO144 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 15 | GPIO143 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 14 | GPIO142 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 13 | GPIO141 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 12 | GPIO140 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 11 | GPIO139 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 10 | GPIO138 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 9 | GPIO137 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 8 | GPIO136 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 7 | GPIO135 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 6 | GPIO134 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 5 | GPIO133 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 4 | GPIO132 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 3 | GPIO131 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 2 | GPIO130 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 1 | GPIO129 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 0 | GPIO128 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.3.21 GPFDAT Register (Offset = 28h) [Reset = 00000000h]

GPFDAT is shown in Figure 8-122 and described in Table 8-134.

Return to the Summary Table.

GPIO F Data Register (GPIO160 to 168)
Reading this register reflects the current state of the GPIO pin regardless of which mode the GPIO is in.
Writing to this register will set the GPIO pin high or low if the pin is enabled for GPIO output mode. If the GPIO is not in output mode the value written is latched but will not be reflected on the GPIO pin or reads of the GPxDAT register. The written value latched will become active when the GPIO is put into GPIO Output mode
A system reset will clear all bits and latched values to zero.
NOTE: Bit-wise read-modify-write operations should not be performed on this register. For bit-wise operations the GPxSET, GPxCLEAR, or GPxTOGGLE registers should be used instead. If direct writes to GPxDAT are necessary, the entire register should be written at one time.

**Figure 8-122. GPFDAT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | 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 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | 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 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | GPIO168 |
| 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 |
| GPIO167 | GPIO166 | GPIO165 | GPIO164 | GPIO163 | GPIO162 | GPIO161 | GPIO160 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 8-134. GPFDAT 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 |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-134. GPFDAT 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 | GPIO168 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 7 | GPIO167 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 6 | GPIO166 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 5 | GPIO165 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 4 | GPIO164 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 3 | GPIO163 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 2 | GPIO162 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 1 | GPIO161 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |
| 0 | GPIO160 | R/W | 0h | Data Register for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.3.22 GPFSET Register (Offset = 2Ah) [Reset = 00000000h]

GPFSET is shown in Figure 8-123 and described in Table 8-135.

Return to the Summary Table.

GPIO F Data Set Register (GPIO160 to 168)
Writing a 1 will force GPIO output data latch to 1.
Writes of 0 are ignored.
Always reads back a 0.

**Figure 8-123. GPFSET Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | 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 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | 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 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | GPIO168 |
| 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 |
| GPIO167 | GPIO166 | GPIO165 | GPIO164 | GPIO163 | GPIO162 | GPIO161 | GPIO160 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 8-135. GPFSET 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 |
| 9 | RESERVED | R/W | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-135. GPFSET Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 8 | GPIO168 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 7 | GPIO167 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 6 | GPIO166 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 5 | GPIO165 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 4 | GPIO164 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 3 | GPIO163 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 2 | GPIO162 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 1 | GPIO161 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |
| 0 | GPIO160 | R/W | 0h | Output Set bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.3.23 GPFCLEAR Register (Offset = 2Ch) [Reset = 00000000h]

GPFCLEAR is shown in Figure 8-124 and described in Table 8-136.

Return to the Summary Table.

GPIO F Data Clear Register (GPIO160 to 168)
Writing a 1 will force GPIO0 output data latch to 0.
Writes of 0 are ignored.
Always reads back a 0.

**Figure 8-124. GPFCLEAR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | 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 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | 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 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | GPIO168 |
| 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 |
| GPIO167 | GPIO166 | GPIO165 | GPIO164 | GPIO163 | GPIO162 | GPIO161 | GPIO160 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 8-136. GPFCLEAR 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 |
| 9 | RESERVED | R/W | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-136. GPFCLEAR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 8 | GPIO168 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 7 | GPIO167 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 6 | GPIO166 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 5 | GPIO165 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 4 | GPIO164 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 3 | GPIO163 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 2 | GPIO162 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 1 | GPIO161 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |
| 0 | GPIO160 | R/W | 0h | Output Clear bit for this pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 8.10.3.24 GPFTOGGLE Register (Offset = 2Eh) [Reset = 00000000h]

GPFTOGGLE is shown in Figure 8-125 and described in Table 8-137.

Return to the Summary Table.

GPIO F Data Toggle Register (GPIO160 to 168)
Writing a 1 will toggle GPIO0 output data latch 1 to 0 or 0 to 1.
Writes of 0 are ignored.
Always reads back a 0.

**Figure 8-125. GPFTOGGLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | 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 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | 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 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | GPIO168 |
| 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 |
| GPIO167 | GPIO166 | GPIO165 | GPIO164 | GPIO163 | GPIO162 | GPIO161 | GPIO160 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 8-137. GPFTOGGLE 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 |
| 9 | RESERVED | R/W | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-137. GPFTOGGLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 8 | GPIO168 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 7 | GPIO167 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 6 | GPIO166 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 5 | GPIO165 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 4 | GPIO164 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 3 | GPIO163 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 2 | GPIO162 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 1 | GPIO161 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |
| 0 | GPIO160 | R/W | 0h | Output Toggle Register GPIO pin<br>Reset type: SYSRSn |

### *8.10.4 GPIO Registers to Driverlib Functions*

**Table 8-138. GPIO Registers to Driverlib Functions**

| File | Driverlib Function |
|---|---|
| **GPACTRL** | |
| gpio.c | GPIO_setQualificationPeriod |
| **GPAQSEL1** | |
| gpio.c | GPIO_setQualificationMode |
| gpio.c | GPIO_getQualificationMode |
| **GPAQSEL2** | |
| - | See GPAQSEL1 |
| **GPAMUX1** | |
| gpio.c | GPIO_setPinConfig |
| **GPAMUX2** | |
| - | See GPAMUX1 |
| **GPADIR** | |
| gpio.c | GPIO_setDirectionMode |
| gpio.c | GPIO_getDirectionMode |
| **GPAPUD** | |
| gpio.c | GPIO_setPadConfig |
| gpio.c | GPIO_getPadConfig |
| **GPAINV** | |
| gpio.c | GPIO_setPadConfig |
| gpio.c | GPIO_getPadConfig |
| **GPAODR** | |
| gpio.c | GPIO_setPadConfig |
| gpio.c | GPIO_getPadConfig |
| **GPAGMUX1** | |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-138. GPIO Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| gpio.c | GPIO_setPinConfig |
| **GPAGMUX2** | |
| - | See GPAGMUX1 |
| **GPACSEL1** | |
| gpio.c | GPIO_setControllerCore |
| **GPACSEL2** | |
| - | See GPACSEL1 |
| **GPACSEL3** | |
| - | See GPACSEL1 |
| **GPACSEL4** | |
| - | See GPACSEL1 |
| **GPALOCK** | |
| gpio.h | GPIO_lockPortConfig |
| gpio.h | GPIO_unlockPortConfig |
| **GPACR** | |
| gpio.h | GPIO_commitPortConfig |
| **GPBCTRL** | |
| - | See GPACTRL |
| **GPBQSEL1** | |
| - | See GPAQSEL1 |
| **GPBQSEL2** | |
| - | See GPAQSEL1 |
| **GPBMUX1** | |
| - | See GPAMUX1 |
| **GPBMUX2** | |
| - | See GPAMUX1 |
| **GPBDIR** | |
| - | See GPADIR |
| **GPBPUD** | |
| - | See GPAPUD |
| **GPBINV** | |
| - | See GPAINV |
| **GPBODR** | |
| - | See GPAODR |
| **GPBAMSEL** | |
| gpio.c | GPIO_setAnalogMode |
| **GPBGMUX1** | |
| - | See GPAGMUX1 |
| **GPBGMUX2** | |
| - | See GPAGMUX1 |
| **GPBCSEL1** | |
| - | See GPACSEL1 |
| **GPBCSEL2** | |
| - | See GPACSEL1 |
| **GPBCSEL3** | |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-138. GPIO Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| - | See GPACSEL1 |
| **GPBCSEL4** | |
| - | See GPACSEL1 |
| **GPBLOCK** | |
| - | See GPALOCK |
| **GPBCR** | |
| - | See GPACR |
| **GPCCTRL** | |
| - | See GPACTRL |
| **GPCQSEL1** | |
| - | See GPAQSEL1 |
| **GPCQSEL2** | |
| - | See GPAQSEL1 |
| **GPCMUX1** | |
| - | See GPAMUX1 |
| **GPCMUX2** | |
| - | See GPAMUX1 |
| **GPCDIR** | |
| - | See GPADIR |
| **GPCPUD** | |
| - | See GPAPUD |
| **GPCINV** | |
| - | See GPAINV |
| **GPCODR** | |
| - | See GPAODR |
| **GPCGMUX1** | |
| - | See GPAGMUX1 |
| **GPCGMUX2** | |
| - | See GPAGMUX1 |
| **GPCCSEL1** | |
| - | See GPACSEL1 |
| **GPCCSEL2** | |
| - | See GPACSEL1 |
| **GPCCSEL3** | |
| - | See GPACSEL1 |
| **GPCCSEL4** | |
| - | See GPACSEL1 |
| **GPCLOCK** | |
| - | See GPALOCK |
| **GPCCR** | |
| - | See GPACR |
| **GPDCTRL** | |
| - | See GPACTRL |
| **GPDQSEL1** | |
| - | See GPAQSEL1 |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-138. GPIO Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| **GPDQSEL2** | |
| - | See GPAQSEL1 |
| **GPDMUX1** | |
| - | See GPAMUX1 |
| **GPDMUX2** | |
| - | See GPAMUX1 |
| **GPDDIR** | |
| - | See GPADIR |
| **GPDPUD** | |
| - | See GPAPUD |
| **GPDINV** | |
| - | See GPAINV |
| **GPDODR** | |
| - | See GPAODR |
| **GPDGMUX1** | |
| - | See GPAGMUX1 |
| **GPDGMUX2** | |
| - | See GPAGMUX1 |
| **GPDCSEL1** | |
| - | See GPACSEL1 |
| **GPDCSEL2** | |
| - | See GPACSEL1 |
| **GPDCSEL3** | |
| - | See GPACSEL1 |
| **GPDCSEL4** | |
| - | See GPACSEL1 |
| **GPDLOCK** | |
| - | See GPALOCK |
| **GPDCR** | |
| - | See GPACR |
| **GPECTRL** | |
| - | See GPACTRL |
| **GPEQSEL1** | |
| - | See GPAQSEL1 |
| **GPEQSEL2** | |
| - | See GPAQSEL1 |
| **GPEMUX1** | |
| - | See GPAMUX1 |
| **GPEMUX2** | |
| - | See GPAMUX1 |
| **GPEDIR** | |
| - | See GPADIR |
| **GPEPUD** | |
| - | See GPAPUD |
| **GPEINV** | |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-138. GPIO Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| - | See GPAINV |
| **GPEODR** | |
| - | See GPAODR |
| **GPEGMUX1** | |
| - | See GPAGMUX1 |
| **GPEGMUX2** | |
| - | See GPAGMUX1 |
| **GPECSEL1** | |
| - | See GPACSEL1 |
| **GPECSEL2** | |
| - | See GPACSEL1 |
| **GPECSEL3** | |
| - | See GPACSEL1 |
| **GPECSEL4** | |
| - | See GPACSEL1 |
| **GPELOCK** | |
| - | See GPALOCK |
| **GPECR** | |
| - | See GPACR |
| **GPFCTRL** | |
| - | See GPACTRL |
| **GPFQSEL1** | |
| - | See GPAQSEL1 |
| **GPFMUX1** | |
| - | See GPAMUX1 |
| **GPFDIR** | |
| - | See GPADIR |
| **GPFPUD** | |
| - | See GPAPUD |
| **GPFINV** | |
| - | See GPAINV |
| **GPFODR** | |
| - | See GPAODR |
| **GPFGMUX1** | |
| - | See GPAGMUX1 |
| **GPFCSEL1** | |
| - | See GPACSEL1 |
| **GPFCSEL2** | |
| - | See GPACSEL1 |
| **GPFLOCK** | |
| - | See GPALOCK |
| **GPFCR** | |
| - | See GPACR |
| **GPADAT** | |
| gpio.h | GPIO_readPin |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-138. GPIO Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| gpio.h | GPIO_readPortData |
| gpio.h | GPIO_writePortData |
| **GPASET** | |
| gpio.h | GPIO_writePin |
| gpio.h | GPIO_setPortPins |
| **GPACLEAR** | |
| gpio.h | GPIO_writePin |
| gpio.h | GPIO_clearPortPins |
| **GPATOGGLE** | |
| gpio.h | GPIO_togglePin |
| gpio.h | GPIO_togglePortPins |
| **GPBDAT** | |
| - | See GPADAT |
| **GPBSET** | |
| - | See GPASET |
| **GPBCLEAR** | |
| - | See GPACLEAR |
| **GPBTOGGLE** | |
| - | See GPATOGGLE |
| **GPCDAT** | |
| - | See GPADAT |
| **GPCSET** | |
| - | See GPASET |
| **GPCCLEAR** | |
| - | See GPACLEAR |
| **GPCTOGGLE** | |
| - | See GPATOGGLE |
| **GPDDAT** | |
| - | See GPADAT |
| **GPDSET** | |
| - | See GPASET |
| **GPDCLEAR** | |
| - | See GPACLEAR |
| **GPDTOGGLE** | |
| - | See GPATOGGLE |
| **GPEDAT** | |
| - | See GPADAT |
| **GPESET** | |
| - | See GPASET |
| **GPECLEAR** | |
| - | See GPACLEAR |
| **GPETOGGLE** | |
| - | See GPATOGGLE |
| **GPFDAT** | |
| - | See GPADAT |



Copyright © 2024 Texas Instruments Incorporated

**Table 8-138. GPIO Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| **GPFSET** | |
| - | See GPASET |
| **GPFCLEAR** | |
| - | See GPACLEAR |
| **GPFTOGGLE** | |
| - | See GPATOGGLE |



Copyright © 2024 Texas Instruments Incorporated

Chapter 9

# Crossbar (X-BAR)

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 four X-BARs:

- Input X-BAR
- Output X-BAR
- CLB 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 CLB X-BAR and ePWM X-BAR take signals to the CLB and ePWM modules, respectively.

**9.1 Input X-BAR** ..... 1230
**9.2 ePWM, CLB, and GPIO Output X-BAR** ..... 1233
**9.3 XBAR Registers** ..... 1241



Copyright © 2024 Texas Instruments Incorporated

## 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.



Copyright © 2024 Texas Instruments Incorporated

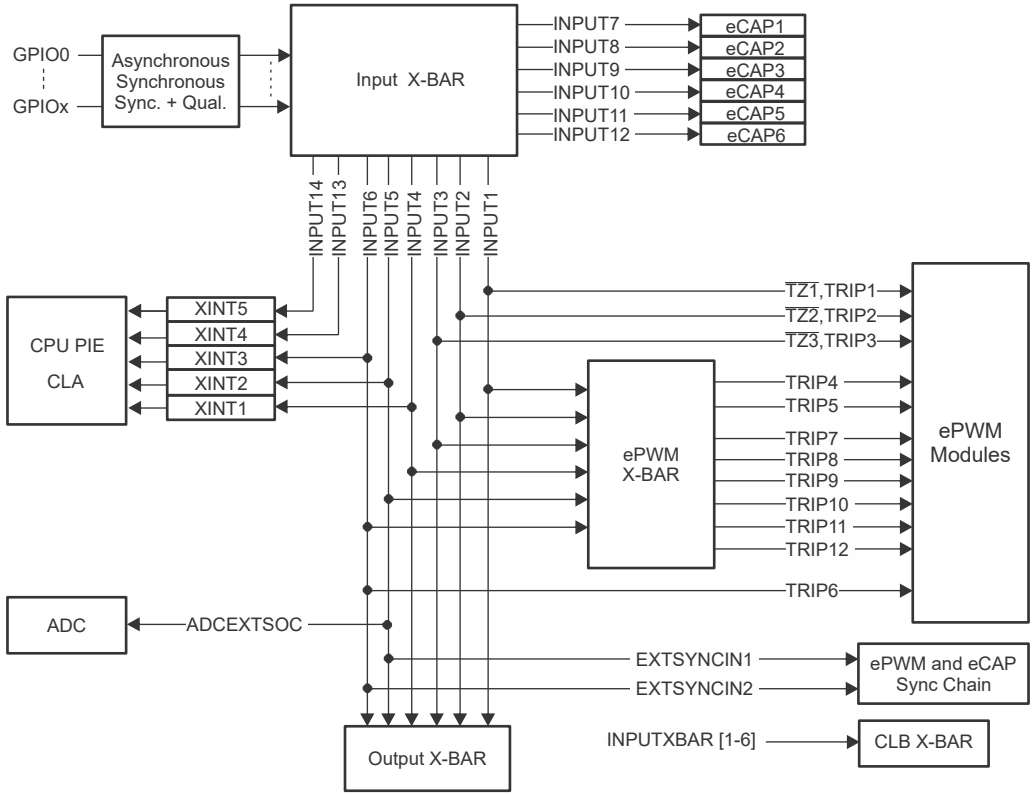



**Figure 9-1. Input X-BAR**

**Note**

INPUTXBARx, INPUTXBAR_INPUTx, and INPUTx (when referenced in the context of Input X-BAR) are equivalent in all C2000 software and documentation.



Copyright © 2024 Texas Instruments Incorporated

**Table 9-1. Input X-BAR Destinations**

| INPUT | ECAP / HRCAP | EPWM X-BAR | CLB X-BAR | OUTPUT X-BAR | CPU XINT | EPWM TRIP | ADC START OF CONVERSION | EPWM / ECAP SYNC |
|---|---|---|---|---|---|---|---|---|
| 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 | ECAP1 | - | - | - | - | - | - | - |
| 8 | ECAP2 | - | - | - | - | - | - | - |
| 9 | ECAP3 | - | - | - | - | - | - | - |
| 10 | ECAP4 | - | - | - | - | - | - | - |
| 11 | ECAP5 | - | - | - | - | - | - | - |
| 12 | ECAP6 | - | - | - | - | - | - | - |
| 13 | - | - | - | - | XINT4 | - | - | - |
| 14 | - | - | - | - | XINT5 | - | - | - |



Copyright © 2024 Texas Instruments Incorporated

## 9.2 ePWM, CLB, and GPIO Output X-BAR

This section describes the ePWM, CLB, and GPIO Output X-BAR. .

### *9.2.1 ePWM X-BAR*

The ePWM X-BAR brings signals to the ePWM modules. Specifically, the ePWM X-BAR is connected to the Digital Compare (DC) submodule of each ePWM module for actions such as tripzones and syncing. 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 TRIPOUTINV register.

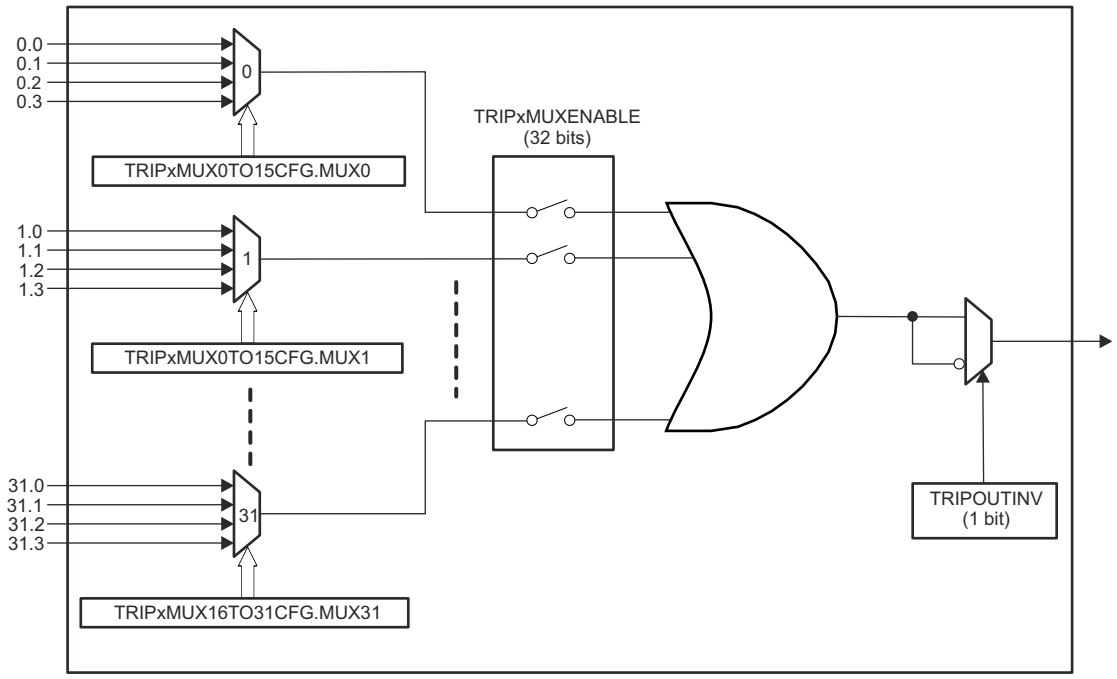



**Figure 9-2. ePWM X-BAR Architecture - Single Output**



Copyright © 2024 Texas Instruments Incorporated

**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 | CLB1_OUT12 | ADCCEVT1 |
| G2 | CMPSS2_CTRIPH | CMPSS2_CTRIPH_OR_CTRIPL | ADCAEVT2 | ECAP2_OUT |
| G3 | CMPSS2_CTRIPL | INPUTXBAR2 | CLB1_OUT13 | ADCCEVT2 |
| G4 | CMPSS3_CTRIPH | CMPSS3_CTRIPH_OR_CTRIPL | ADCAEVT3 | ECAP3_OUT |
| G5 | CMPSS3_CTRIPL | INPUTXBAR3 | CLB2_OUT12 | ADCCEVT3 |
| G6 | CMPSS4_CTRIPH | CMPSS4_CTRIPH_OR_CTRIPL | ADCAEVT4 | ECAP4_OUT |
| G7 | CMPSS4_CTRIPL | INPUTXBAR4 | CLB2_OUT13 | ADCCEVT4 |
| G8 | CMPSS5_CTRIPH | CMPSS5_CTRIPH_OR_CTRIPL | ADCBEVT1 | ECAP5_OUT |
| G9 | CMPSS5_CTRIPL | INPUTXBAR5 | CLB3_OUT12 | ADCDEVT1 |
| G10 | CMPSS6_CTRIPH | CMPSS6_CTRIPH_OR_CTRIPL | ADCBEVT2 | ECAP6_OUT |
| G11 | CMPSS6_CTRIPL | INPUTXBAR6 | CLB3_OUT13 | ADCDEVT2 |
| G12 | CMPSS7_CTRIPH | CMPSS7_CTRIPH_OR_CTRIPL | ADCBEVT3 | Reserved |
| G13 | CMPSS7_CTRIPL | ADCSOCAO | CLB4_OUT12 | ADCDEVT3 |
| G14 | CMPSS8_CTRIPH | CMPSS8_CTRIPH_OR_CTRIPL | ADCBEVT4 | EXTSYNCOUT |
| G15 | CMPSS8_CTRIPL | ADCSOCBO | CLB4_OUT13 | ADCDEVT4 |
| G16 | SD1FLT1_COMPH | SD1FLT1_COMPH_OR_COMPL | Reserved | Reserved |
| G17 | SD1FLT1_COMPL | Reserved | Reserved | Reserved |
| G18 | SD1FLT2_COMPH | SD1FLT2_COMPH_OR_COMPL | Reserved | Reserved |
| G19 | SD1FLT2_COMPL | Reserved | Reserved | Reserved |
| G20 | SD1FLT3_COMPH | SD1FLT3_COMPH_OR_COMPL | Reserved | Reserved |
| G21 | SD1FLT3_COMPL | Reserved | Reserved | Reserved |
| G22 | SD1FLT4_COMPH | SD1FLT4_COMPH_OR_COMPL | Reserved | Reserved |
| G23 | SD1FLT4_COMPL | Reserved | Reserved | Reserved |
| G24 | SD2FLT1_COMPH | SD2FLT1_COMPH_OR_COMPL | Reserved | Reserved |
| G25 | SD2FLT1_COMPL | Reserved | Reserved | Reserved |
| G26 | SD2FLT2_COMPH | SD2FLT2_COMPH_OR_COMPL | Reserved | Reserved |
| G27 | SD2FLT2_COMPL | Reserved | Reserved | Reserved |
| G28 | SD2FLT3_COMPH | SD2FLT3_COMPH_OR_COMPL | Reserved | Reserved |
| G29 | SD2FLT3_COMPL | Reserved | Reserved | Reserved |
| G30 | SD2FLT4_COMPH | SD2FLT4_COMPH_OR_COMPL | Reserved | Reserved |
| G31 | SD2FLT4_COMPL | Reserved | Reserved | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 9.2.2 CLB X-BAR

The CLB X-BAR brings signals to the CLB modules. Figure 9-3 shows the architecture of the CLB X-BAR. Note that the architecture of the CLB X-BAR is identical to the architecture of the GPIO Output X-BAR (with the exception of the output latch).

#### 9.2.2.1 CLB X-BAR Architecture

The CLB X-BAR has eight outputs that are routed to each CLB module. Figure 9-3 represents the architecture of a single output, but the output is identical to the architecture of all of the other outputs.

First, determine the signals that can be passed to the CLB by referencing Table 9-3. Select up to one signal per mux (31 total muxes) for each AUXSIGx output. Select the inputs to each mux using the AUXSIGxMUX0TO15CFG and AUXSIGxMUX16TO31CFG registers. To pass any signal through to the CLB, enable the mux in the AUXSIGxMUXENABLE register. All muxes that are enabled are logically ORed before being passed on to the respective AUXSIGx signal on the CLB. To optionally invert the signal, use the AUXSIGOUTINV register.

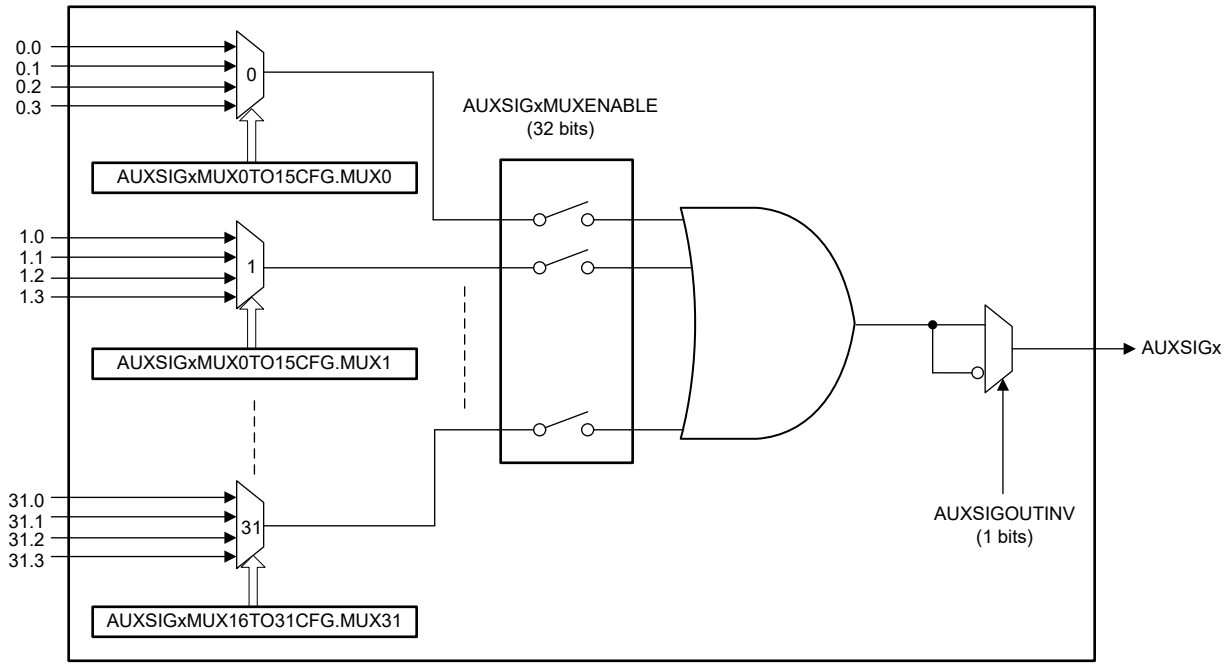



**Figure 9-3. CLB X-BAR Architecture - Single Output**



Copyright © 2024 Texas Instruments Incorporated

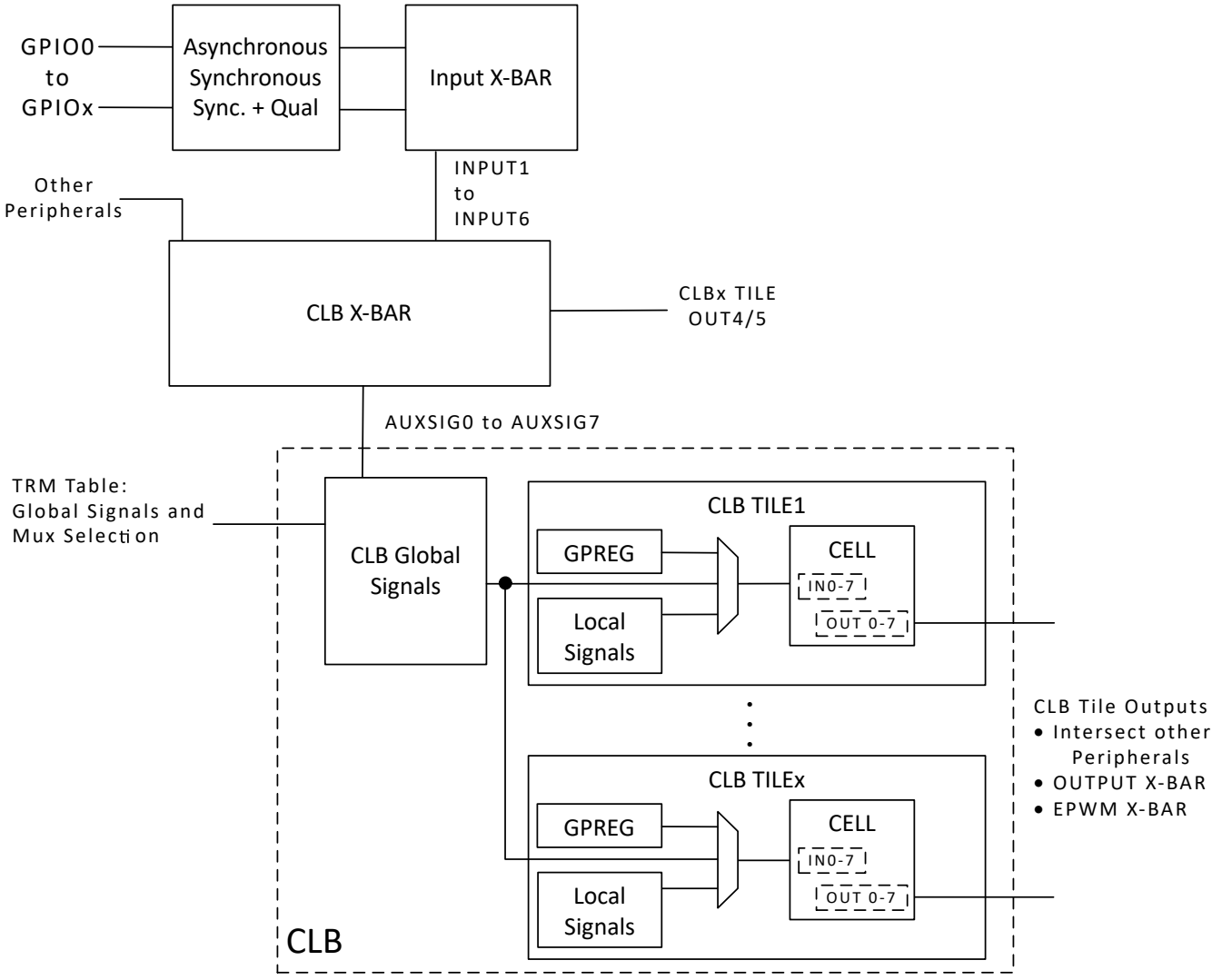



**Figure 9-4. GPIO to CLB Tile Connections**



Copyright © 2024 Texas Instruments Incorporated

**Table 9-3. CLB X-BAR Mux Configuration Table**

| Mux | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| G0 | CMPSS1_CTRIPH | CMPSS1_CTRIPH_OR_CTRIPL | ADCAEVT1 | ECAP1_OUT |
| G1 | CMPSS1_CTRIPL | INPUTXBAR1 | CLB1_OUT12 | ADCCEVT1 |
| G2 | CMPSS2_CTRIPH | CMPSS2_CTRIPH_OR_CTRIPL | ADCAEVT2 | ECAP2_OUT |
| G3 | CMPSS2_CTRIPL | INPUTXBAR2 | CLB1_OUT13 | ADCCEVT2 |
| G4 | CMPSS3_CTRIPH | CMPSS3_CTRIPH_OR_CTRIPL | ADCAEVT3 | ECAP3_OUT |
| G5 | CMPSS3_CTRIPL | INPUTXBAR3 | CLB2_OUT12 | ADCCEVT3 |
| G6 | CMPSS4_CTRIPH | CMPSS4_CTRIPH_OR_CTRIPL | ADCAEVT4 | ECAP4_OUT |
| G7 | CMPSS4_CTRIPL | INPUTXBAR4 | CLB2_OUT13 | ADCCEVT4 |
| G8 | CMPSS5_CTRIPH | CMPSS5_CTRIPH_OR_CTRIPL | ADCBEVT1 | ECAP5_OUT |
| G9 | CMPSS5_CTRIPL | INPUTXBAR5 | CLB3_OUT12 | ADCDEVT1 |
| G10 | CMPSS6_CTRIPH | CMPSS6_CTRIPH_OR_CTRIPL | ADCBEVT2 | ECAP6_OUT |
| G11 | CMPSS6_CTRIPL | INPUTXBAR6 | CLB3_OUT13 | ADCDEVT2 |
| G12 | CMPSS7_CTRIPH | CMPSS7_CTRIPH_OR_CTRIPL | ADCBEVT3 | Reserved |
| G13 | CMPSS7_CTRIPL | ADCSOCAO | CLB4_OUT12 | ADCDEVT3 |
| G14 | CMPSS8_CTRIPH | CMPSS8_CTRIPH_OR_CTRIPL | ADCBEVT4 | EXTSYNCOUT |
| G15 | CMPSS8_CTRIPL | ADCSOCBO | CLB4_OUT13 | ADCDEVT4 |
| G16 | SD1FLT1_COMPH | SD1FLT1_COMPH_OR_COMPL | Reserved | Reserved |
| G17 | SD1FLT1_COMPL | Reserved | Reserved | Reserved |
| G18 | SD1FLT2_COMPH | SD1FLT2_COMPH_OR_COMPL | Reserved | Reserved |
| G19 | SD1FLT2_COMPL | Reserved | Reserved | Reserved |
| G20 | SD1FLT3_COMPH | SD1FLT3_COMPH_OR_COMPL | Reserved | Reserved |
| G21 | SD1FLT3_COMPL | Reserved | Reserved | Reserved |
| G22 | SD1FLT4_COMPH | SD1FLT4_COMPH_OR_COMPL | Reserved | Reserved |
| G23 | SD1FLT4_COMPL | Reserved | Reserved | Reserved |
| G24 | SD2FLT1_COMPH | SD2FLT1_COMPH_OR_COMPL | Reserved | Reserved |
| G25 | SD2FLT1_COMPL | Reserved | Reserved | Reserved |
| G26 | SD2FLT2_COMPH | SD2FLT2_COMPH_OR_COMPL | Reserved | Reserved |
| G27 | SD2FLT2_COMPL | Reserved | Reserved | Reserved |
| G28 | SD2FLT3_COMPH | SD2FLT3_COMPH_OR_COMPL | Reserved | Reserved |
| G29 | SD2FLT3_COMPL | Reserved | Reserved | Reserved |
| G30 | SD2FLT4_COMPH | SD2FLT4_COMPH_OR_COMPL | Reserved | Reserved |
| G31 | SD2FLT4_COMPL | Reserved | Reserved | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 9.2.3 GPIO Output X-BAR

The GPIO Output X-BAR takes signals from inside the device and brings them out to a GPIO. Figure 9-5 shows the architecture of the GPIO Output X-BAR. The 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 input or a logical-OR of many inputs.

#### 9.2.3.1 GPIO Output X-BAR Architecture

The GPIO Output X-BAR has eight outputs that are routed to the GPIO module. Figure 9-5 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 9-4. 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 final output is only recognized on the GPIO if the proper OUTPUTx muxing options are selected using the GPIO registers.

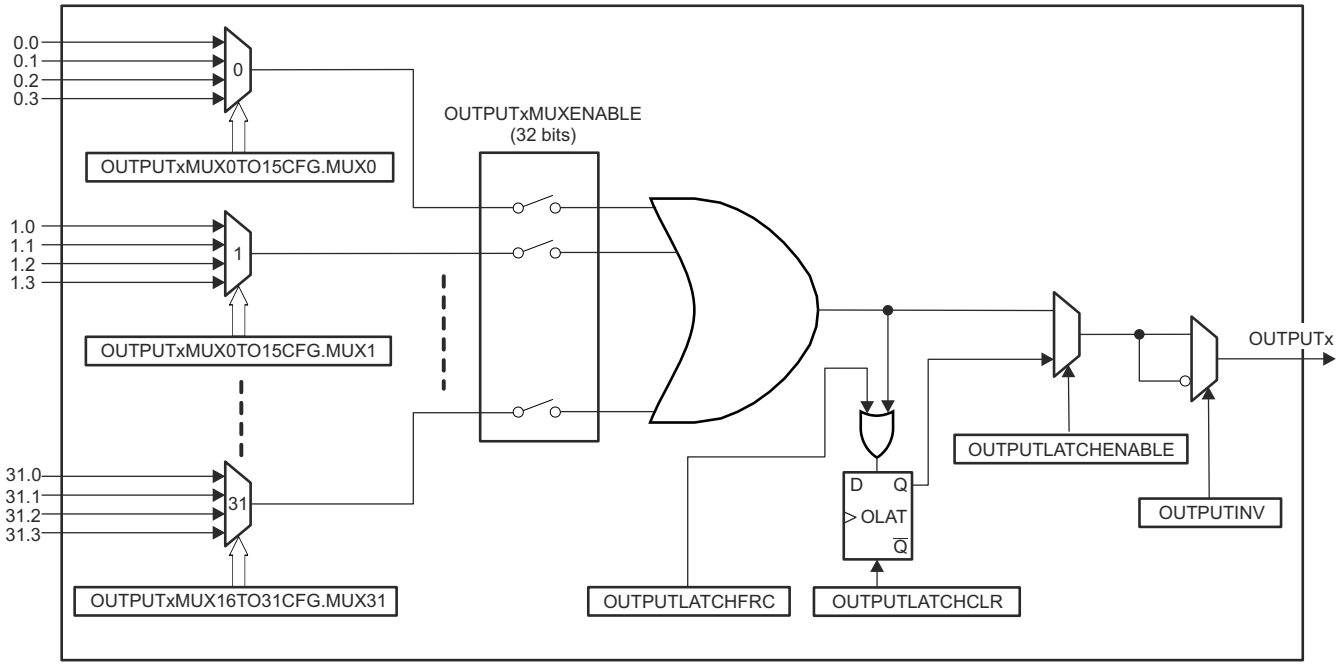



**Figure 9-5. GPIO Output X-BAR Architecture**

**Note**

Do not use "Reserved" signals in your application.

The ADCSOCAO and ADCSOCBO signals are active-high when routed through the X-BAR. The signal can be inverted by the respective OUTPUTINV bit depending on the application.



Copyright © 2024 Texas Instruments Incorporated

**Table 9-4. Output X-BAR Mux Configuration Table**

| Mux | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| G0 | CMPSS1_CTRIPOUTH | CMPSS1_CTRIPOUTOUTH_OR_CTRIPOUTOUTL | ADCAEVT1 | ECAP1_OUT |
| G1 | CMPSS1_CTRIPOUTL | INPUTXBAR1 | CLB1_OUT12 | ADCCEVT1 |
| G2 | CMPSS2_CTRIPOUTH | CMPSS2_CTRIPOUTOUTH_OR_CTRIPOUTOUTL | ADCAEVT2 | ECAP2_OUT |
| G3 | CMPSS2_CTRIPOUTL | INPUTXBAR2 | CLB1_OUT13 | ADCCEVT2 |
| G4 | CMPSS3_CTRIPOUTH | CMPSS3_CTRIPOUTOUTH_OR_CTRIPOUTOUTL | ADCAEVT3 | ECAP3_OUT |
| G5 | CMPSS3_CTRIPOUTL | INPUTXBAR3 | CLB2_OUT12 | ADCCEVT3 |
| G6 | CMPSS4_CTRIPOUTH | CMPSS4_CTRIPOUTOUTH_OR_CTRIPOUTOUTL | ADCAEVT4 | ECAP4_OUT |
| G7 | CMPSS4_CTRIPOUTL | INPUTXBAR4 | CLB2_OUT13 | ADCCEVT4 |
| G8 | CMPSS5_CTRIPOUTH | CMPSS5_CTRIPOUTOUTH_OR_CTRIPOUTOUTL | ADCBEVT1 | ECAP5_OUT |
| G9 | CMPSS5_CTRIPOUTL | INPUTXBAR5 | CLB3_OUT12 | ADCDEVT1 |
| G10 | CMPSS6_CTRIPOUTH | CMPSS6_CTRIPOUTOUTH_OR_CTRIPOUTOUTL | ADCBEVT2 | ECAP6_OUT |
| G11 | CMPSS6_CTRIPOUTL | INPUTXBAR6 | CLB3_OUT13 | ADCDEVT2 |
| G12 | CMPSS7_CTRIPOUTH | CMPSS7_CTRIPOUTOUTH_OR_CTRIPOUTOUTL | ADCBEVT3 | Reserved |
| G13 | CMPSS7_CTRIPOUTL | ADCSOCAO | CLB4_OUT12 | ADCDEVT3 |
| G14 | CMPSS8_CTRIPOUTH | CMPSS8_CTRIPOUTOUTH_OR_CTRIPOUTOUTL | ADCBEVT4 | EXTSYNCOUT |
| G15 | CMPSS8_CTRIPOUTL | ADCSOCBO | CLB4_OUT13 | ADCDEVT4 |
| G16 | SD1FLT1_COMPH | SD1FLT1_COMPH_OR_COMPL | Reserved | Reserved |
| G17 | SD1FLT1_COMPL | Reserved | Reserved | Reserved |
| G18 | SD1FLT2_COMPH | SD1FLT2_COMPH_OR_COMPL | Reserved | Reserved |
| G19 | SD1FLT2_COMPL | Reserved | Reserved | Reserved |
| G20 | SD1FLT3_COMPH | SD1FLT3_COMPH_OR_COMPL | Reserved | Reserved |
| G21 | SD1FLT3_COMPL | Reserved | Reserved | Reserved |
| G22 | SD1FLT4_COMPH | SD1FLT4_COMPH_OR_COMPL | Reserved | Reserved |
| G23 | SD1FLT4_COMPL | Reserved | Reserved | Reserved |
| G24 | SD2FLT1_COMPH | SD2FLT1_COMPH_OR_COMPL | Reserved | Reserved |
| G25 | SD2FLT1_COMPL | Reserved | Reserved | Reserved |
| G26 | SD2FLT2_COMPH | SD2FLT2_COMPH_OR_COMPL | Reserved | Reserved |
| G27 | SD2FLT2_COMPL | Reserved | Reserved | Reserved |
| G28 | SD2FLT3_COMPH | SD2FLT3_COMPH_OR_COMPL | Reserved | Reserved |
| G29 | SD2FLT3_COMPL | Reserved | Reserved | Reserved |
| G30 | SD2FLT4_COMPH | SD2FLT4_COMPH_OR_COMPL | Reserved | Reserved |
| G31 | SD2FLT4_COMPL | Reserved | Reserved | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 9.2.4 X-BAR Flags

With the exception of the CMPSS signals, the ePWM X-BAR and the Output X-BAR have all of the same input signals. Due to the inputs being similar between the ePWM X-BAR, CLB 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 9-6 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 XBARCLRx register.

**Note**

Not all input sources are routed to all X-BAR modules. Refer to the X-BAR specific configuration tables for exact connections.

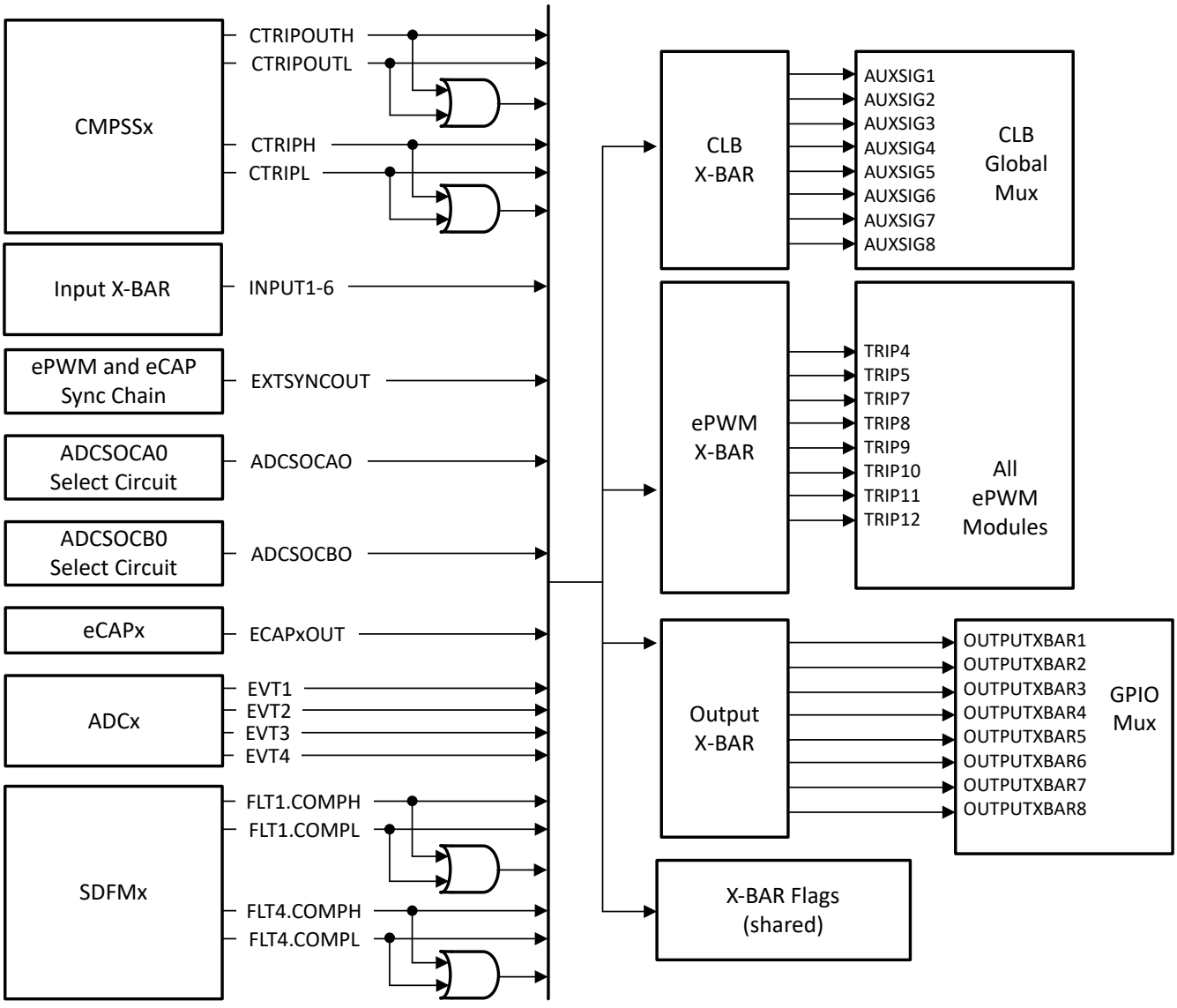



**Figure 9-6. X-BAR Input Sources**



Copyright © 2024 Texas Instruments Incorporated

## 9.3 XBAR Registers

This section describes the Crossbar registers.

### *9.3.1 XBAR Base Addresses*

**Table 9-5. XBAR Base Address Table**

| Device Registers | Register Name | Start Address | End Address |
|---|---|---|---|
| InputXbarRegs(1) | INPUT_XBAR_REGS | 0x0000_7900 | 0x0000_791F |
| XbarRegs(1) | XBAR_REGS | 0x0000 7920 | 0x0000_793F |
| EPwmXbarRegs(1) | ePWM_XBAR_REGS | 0x0000_7A00 | 0x0000_7A3F |
| ClbXbarRegs(1) | CLB_XBAR_REGS | 0x0000_7A40 | 0x0000_7A7F |
| OutputXbarRegs(1) | OUTPUT_XBAR_REGS | 0x0000_7A80 | 0x0000_7ABF |

(1) Only available on CPU1.



Copyright © 2024 Texas Instruments Incorporated

### 9.3.2 INPUT_XBAR_REGS Registers

Table 9-6 lists the memory-mapped registers for the INPUT_XBAR_REGS registers. All register offset addresses not listed in Table 9-6 should be considered as reserved locations and the register contents should not be modified.

**Table 9-6. INPUT_XBAR_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | INPUT1SELECT | INPUT1 Input Select Register (GPIO0 to x) | EALLOW | Go |
| 1h | INPUT2SELECT | INPUT2 Input Select Register (GPIO0 to x) | EALLOW | Go |
| 2h | INPUT3SELECT | INPUT3 Input Select Register (GPIO0 to x) | EALLOW | Go |
| 3h | INPUT4SELECT | INPUT4 Input Select Register (GPIO0 to x) | EALLOW | Go |
| 4h | INPUT5SELECT | INPUT5 Input Select Register (GPIO0 to x) | EALLOW | Go |
| 5h | INPUT6SELECT | INPUT6 Input Select Register (GPIO0 to x) | EALLOW | Go |
| 6h | INPUT7SELECT | INPUT7 Input Select Register (GPIO0 to x) | EALLOW | Go |
| 7h | INPUT8SELECT | INPUT8 Input Select Register (GPIO0 to x) | EALLOW | Go |
| 8h | INPUT9SELECT | INPUT9 Input Select Register (GPIO0 to x) | EALLOW | Go |
| 9h | INPUT10SELECT | INPUT10 Input Select Register (GPIO0 to x) | EALLOW | Go |
| Ah | INPUT11SELECT | INPUT11 Input Select Register (GPIO0 to x) | EALLOW | Go |
| Bh | INPUT12SELECT | INPUT12 Input Select Register (GPIO0 to x) | EALLOW | Go |
| Ch | INPUT13SELECT | INPUT13 Input Select Register (GPIO0 to x) | EALLOW | Go |
| Dh | INPUT14SELECT | INPUT14 Input Select Register (GPIO0 to x) | EALLOW | Go |
| 1Eh | INPUTSELECTLOCK | Input Select Lock Register | EALLOW | Go |

Complex bit access types are encoded to fit into small table cells. Table 9-7 shows the codes that are used for access types in this section.

**Table 9-7. INPUT_XBAR_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| R-0 | R<br>-0 | Read<br>Returns 0s |
| Write Type | | |
| W | W | Write |
| WSonce | W<br>Sonce | Write<br>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. |



Copyright © 2024 Texas Instruments Incorporated

#### 9.3.2.1 INPUT1SELECT Register (Offset = 0h) [Reset = 0000h]

INPUT1SELECT is shown in Figure 9-7 and described in Table 9-8.

Return to the Summary Table.

INPUT1 Input Select Register (GPIO0 to x)

**Figure 9-7. INPUT1SELECT Register**

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

**Table 9-8. INPUT1SELECT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | SELECT | R/W | 0h | Select GPIO for INPUT1 signal:<br>0x0 : Select GPIO0<br>0x1 : Select GPIO1<br>0x2 : Select GPIO2<br>...<br>0xn : Select GPIOn<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.2.2 INPUT2SELECT Register (Offset = 1h) [Reset = 0000h]

INPUT2SELECT is shown in Figure 9-8 and described in Table 9-9.

Return to the Summary Table.

INPUT2 Input Select Register (GPIO0 to x)

**Figure 9-8. INPUT2SELECT Register**

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

**Table 9-9. INPUT2SELECT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | SELECT | R/W | 0h | Select GPIO for INPUT2 signal:<br>0x0 : Select GPIO0<br>0x1 : Select GPIO1<br>0x2 : Select GPIO2<br>...<br>0xn : Select GPIOn<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.2.3 INPUT3SELECT Register (Offset = 2h) [Reset = 0000h]

INPUT3SELECT is shown in Figure 9-9 and described in Table 9-10.

Return to the Summary Table.

INPUT3 Input Select Register (GPIO0 to x)

**Figure 9-9. INPUT3SELECT Register**

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

**Table 9-10. INPUT3SELECT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | SELECT | R/W | 0h | Select GPIO for INPUT3 signal:<br>0x0 : Select GPIO0<br>0x1 : Select GPIO1<br>0x2 : Select GPIO2<br>...<br>0xn : Select GPIOn<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.2.4 INPUT4SELECT Register (Offset = 3h) [Reset = 0000h]

INPUT4SELECT is shown in Figure 9-10 and described in Table 9-11.

Return to the Summary Table.

INPUT4 Input Select Register (GPIO0 to x)

**Figure 9-10. INPUT4SELECT Register**

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

**Table 9-11. INPUT4SELECT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | SELECT | R/W | 0h | Select GPIO for INPUT4 signal:<br>0x0 : Select GPIO0<br>0x1 : Select GPIO1<br>0x2 : Select GPIO2<br>...<br>0xn : Select GPIOn<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

#### 9.3.2.5 INPUT5SELECT Register (Offset = 4h) [Reset = 0000h]

INPUT5SELECT is shown in Figure 9-11 and described in Table 9-12.

Return to the Summary Table.

INPUT5 Input Select Register (GPIO0 to x)

**Figure 9-11. INPUT5SELECT Register**

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

**Table 9-12. INPUT5SELECT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | SELECT | R/W | 0h | Select GPIO for INPUT5 signal:<br>0x0 : Select GPIO0<br>0x1 : Select GPIO1<br>0x2 : Select GPIO2<br>...<br>0xn : Select GPIOn<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.2.6 INPUT6SELECT Register (Offset = 5h) [Reset = 0000h]

INPUT6SELECT is shown in Figure 9-12 and described in Table 9-13.

Return to the Summary Table.

INPUT6 Input Select Register (GPIO0 to x)

**Figure 9-12. INPUT6SELECT Register**

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

**Table 9-13. INPUT6SELECT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | SELECT | R/W | 0h | Select GPIO for INPUT6 signal:<br>0x0 : Select GPIO0<br>0x1 : Select GPIO1<br>0x2 : Select GPIO2<br>...<br>0xn : Select GPIOn<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.2.7 INPUT7SELECT Register (Offset = 6h) [Reset = 0000h]

INPUT7SELECT is shown in Figure 9-13 and described in Table 9-14.

Return to the Summary Table.

INPUT7 Input Select Register (GPIO0 to x)

**Figure 9-13. INPUT7SELECT Register**

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

**Table 9-14. INPUT7SELECT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | SELECT | R/W | 0h | Select GPIO for INPUT7 signal:<br>0x0 : Select GPIO0<br>0x1 : Select GPIO1<br>0x2 : Select GPIO2<br>...<br>0xn : Select GPIOn<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.2.8 INPUT8SELECT Register (Offset = 7h) [Reset = 0000h]

INPUT8SELECT is shown in Figure 9-14 and described in Table 9-15.

Return to the Summary Table.

INPUT8 Input Select Register (GPIO0 to x)

**Figure 9-14. INPUT8SELECT Register**

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

**Table 9-15. INPUT8SELECT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | SELECT | R/W | 0h | Select GPIO for INPUT8 signal:<br>0x0 : Select GPIO0<br>0x1 : Select GPIO1<br>0x2 : Select GPIO2<br>...<br>0xn : Select GPIOn<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.2.9 INPUT9SELECT Register (Offset = 8h) [Reset = 0000h]

INPUT9SELECT is shown in Figure 9-15 and described in Table 9-16.

Return to the Summary Table.

INPUT9 Input Select Register (GPIO0 to x)

**Figure 9-15. INPUT9SELECT Register**

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

**Table 9-16. INPUT9SELECT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | SELECT | R/W | 0h | Select GPIO for INPUT9 signal:<br>0x0 : Select GPIO0<br>0x1 : Select GPIO1<br>0x2 : Select GPIO2<br>...<br>0xn : Select GPIOn<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.2.10 INPUT10SELECT Register (Offset = 9h) [Reset = 0000h]

INPUT10SELECT is shown in Figure 9-16 and described in Table 9-17.

Return to the Summary Table.

INPUT10 Input Select Register (GPIO0 to x)

**Figure 9-16. INPUT10SELECT Register**

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

**Table 9-17. INPUT10SELECT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | SELECT | R/W | 0h | Select GPIO for INPUT10 signal:<br>0x0 : Select GPIO0<br>0x1 : Select GPIO1<br>0x2 : Select GPIO2<br>...<br>0xn : Select GPIOn<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.2.11 INPUT11SELECT Register (Offset = Ah) [Reset = 0000h]

INPUT11SELECT is shown in Figure 9-17 and described in Table 9-18.

Return to the Summary Table.

INPUT11 Input Select Register (GPIO0 to x)

**Figure 9-17. INPUT11SELECT Register**

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

**Table 9-18. INPUT11SELECT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | SELECT | R/W | 0h | Select GPIO for INPUT11 signal:<br>0x0 : Select GPIO0<br>0x1 : Select GPIO1<br>0x2 : Select GPIO2<br>...<br>0xn : Select GPIOn<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.2.12 INPUT12SELECT Register (Offset = Bh) [Reset = 0000h]

INPUT12SELECT is shown in Figure 9-18 and described in Table 9-19.

Return to the Summary Table.

INPUT12 Input Select Register (GPIO0 to x)

**Figure 9-18. INPUT12SELECT Register**

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

**Table 9-19. INPUT12SELECT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | SELECT | R/W | 0h | Select GPIO for INPUT12 signal:<br>0x0 : Select GPIO0<br>0x1 : Select GPIO1<br>0x2 : Select GPIO2<br>...<br>0xn : Select GPIOn<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.2.13 INPUT13SELECT Register (Offset = Ch) [Reset = 0000h]

INPUT13SELECT is shown in Figure 9-19 and described in Table 9-20.

Return to the Summary Table.

INPUT13 Input Select Register (GPIO0 to x)

**Figure 9-19. INPUT13SELECT Register**

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

**Table 9-20. INPUT13SELECT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | SELECT | R/W | 0h | Select GPIO for INPUT13 signal:<br>0x0 : Select GPIO0<br>0x1 : Select GPIO1<br>0x2 : Select GPIO2<br>...<br>0xn : Select GPIOn<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.2.14 INPUT14SELECT Register (Offset = Dh) [Reset = 0000h]

INPUT14SELECT is shown in Figure 9-20 and described in Table 9-21.

Return to the Summary Table.

INPUT14 Input Select Register (GPIO0 to x)

**Figure 9-20. INPUT14SELECT Register**

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

**Table 9-21. INPUT14SELECT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | SELECT | R/W | 0h | Select GPIO for INPUT14 signal:<br>0x0 : Select GPIO0<br>0x1 : Select GPIO1<br>0x2 : Select GPIO2<br>...<br>0xn : Select GPIOn<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.2.15 INPUTSELECTLOCK Register (Offset = 1Eh) [Reset = 00000000h]

INPUTSELECTLOCK is shown in Figure 9-21 and described in Table 9-22.

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 |
| INPUT16SELECT | INPUT15SELECT | INPUT14SELECT | INPUT13SELECT | INPUT12SELECT | INPUT11SELECT | INPUT10SELECT | INPUT9SELECT |
| R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INPUT8SELECT | INPUT7SELECT | INPUT6SELECT | INPUT5SELECT | INPUT4SELECT | INPUT3SELECT | INPUT2SELECT | INPUT1SELECT |
| R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h |

**Table 9-22. INPUTSELECTLOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15 | INPUT16SELECT | R/WSonce | 0h | Lock bit for INPUT16SELECT Register<br>0: Register is not locked<br>1: Register is locked<br>Reset type: CPU1.SYSRSn |
| 14 | INPUT15SELECT | R/WSonce | 0h | Lock bit for INPUT15SELECT Register<br>0: Register is not locked<br>1: Register is locked<br>Reset type: CPU1.SYSRSn |
| 13 | INPUT14SELECT | R/WSonce | 0h | Lock bit for INPUT14SELECT Register<br>0: Register is not locked<br>1: Register is locked<br>Reset type: CPU1.SYSRSn |
| 12 | INPUT13SELECT | R/WSonce | 0h | Lock bit for INPUT13SELECT Register<br>0: Register is not locked<br>1: Register is locked<br>Reset type: CPU1.SYSRSn |
| 11 | INPUT12SELECT | R/WSonce | 0h | Lock bit for INPUT12SELECT Register<br>0: Register is not locked<br>1: Register is locked<br>Reset type: CPU1.SYSRSn |
| 10 | INPUT11SELECT | R/WSonce | 0h | Lock bit for INPUT11SELECT Register<br>0: Register is not locked<br>1: Register is locked<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-22. INPUTSELECTLOCK Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 9 | INPUT10SELECT | R/WSonce | 0h | Lock bit for INPUT10SELECT Register<br>0: Register is not locked<br>1: Register is locked<br>Reset type: CPU1.SYSRSn |
| 8 | INPUT9SELECT | R/WSonce | 0h | Lock bit for INPUT9SELECT Register<br>0: Register is not locked<br>1: Register is locked<br>Reset type: CPU1.SYSRSn |
| 7 | INPUT8SELECT | R/WSonce | 0h | Lock bit for INPUT8SELECT Register<br>0: Register is not locked<br>1: Register is locked<br>Reset type: CPU1.SYSRSn |
| 6 | INPUT7SELECT | R/WSonce | 0h | Lock bit for INPUT7SELECT Register<br>0: Register is not locked<br>1: Register is locked<br>Reset type: CPU1.SYSRSn |
| 5 | INPUT6SELECT | R/WSonce | 0h | Lock bit for INPUT6SELECT Register<br>0: Register is not locked<br>1: Register is locked<br>Reset type: CPU1.SYSRSn |
| 4 | INPUT5SELECT | R/WSonce | 0h | Lock bit for INPUT5SELECT Register<br>0: Register is not locked<br>1: Register is locked<br>Reset type: CPU1.SYSRSn |
| 3 | INPUT4SELECT | R/WSonce | 0h | Lock bit for INPUT4SELECT Register<br>0: Register is not locked<br>1: Register is locked<br>Reset type: CPU1.SYSRSn |
| 2 | INPUT3SELECT | R/WSonce | 0h | Lock bit for INPUT3SELECT Register<br>0: Register is not locked<br>1: Register is locked<br>Reset type: CPU1.SYSRSn |
| 1 | INPUT2SELECT | R/WSonce | 0h | Lock bit for INPUT2SELECT Register<br>0: Register is not locked<br>1: Register is locked<br>Reset type: CPU1.SYSRSn |
| 0 | INPUT1SELECT | R/WSonce | 0h | Lock bit for INPUT1SELECT Register<br>0: Register is not locked<br>1: Register is locked<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.3 XBAR_REGS Registers

Table 9-23 lists the memory-mapped registers for the XBAR_REGS registers. All register offset addresses not listed in Table 9-23 should be considered as reserved locations and the register contents should not be modified.

**Table 9-23. XBAR_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | XBARFLG1 | X-Bar Input Flag Register 1 | | Go |
| 2h | XBARFLG2 | X-Bar Input Flag Register 2 | | Go |
| 4h | XBARFLG3 | X-Bar Input Flag Register 3 | | Go |
| 8h | XBARCLR1 | X-Bar Input Flag Clear Register 1 | | Go |
| Ah | XBARCLR2 | X-Bar Input Flag Clear Register 2 | | Go |
| Ch | XBARCLR3 | X-Bar Input Flag Clear Register 3 | | Go |

Complex bit access types are encoded to fit into small table cells. Table 9-24 shows the codes that are used for access types in this section.

**Table 9-24. XBAR_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| R-0 | R<br>-0 | Read<br>Returns 0s |
| Write Type | | |
| W1S | W<br>1S | Write<br>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. |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.3.1 XBARFLG1 Register (Offset = 0h) [Reset = 00000000h]

XBARFLG1 is shown in Figure 9-22 and described in Table 9-25.

Return to the Summary Table.

This register is used to flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.
1: Corresponding Input was triggered
0: Corresponding Input was not triggered

**Figure 9-22. XBARFLG1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| CMPSS8_CTRIPOUTH | CMPSS8_CTRIPOUTL | CMPSS7_CTRIPOUTH | CMPSS7_CTRIPOUTL | CMPSS6_CTRIPOUTH | CMPSS6_CTRIPOUTL | CMPSS5_CTRIPOUTH | CMPSS5_CTRIPOUTL |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| CMPSS4_CTRIPOUTH | CMPSS4_CTRIPOUTL | CMPSS3_CTRIPOUTH | CMPSS3_CTRIPOUTL | CMPSS2_CTRIPOUTH | CMPSS2_CTRIPOUTL | CMPSS1_CTRIPOUTH | CMPSS1_CTRIPOUTL |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| CMPSS8_CTRIPH | CMPSS8_CTRIPL | CMPSS7_CTRIPH | CMPSS7_CTRIPL | CMPSS6_CTRIPH | CMPSS6_CTRIPL | CMPSS5_CTRIPH | CMPSS5_CTRIPL |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CMPSS4_CTRIPH | CMPSS4_CTRIPL | CMPSS3_CTRIPH | CMPSS3_CTRIPL | CMPSS2_CTRIPH | CMPSS2_CTRIPL | CMPSS1_CTRIPH | CMPSS1_CTRIPL |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 9-25. XBARFLG1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | CMPSS8_CTRIPOUTH | R | 0h | CMPSS8_CTRIPOUTH X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 30 | CMPSS8_CTRIPOUTL | R | 0h | CMPSS8_CTRIPOUTL X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 29 | CMPSS7_CTRIPOUTH | R | 0h | CMPSS7_CTRIPOUTH X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 28 | CMPSS7_CTRIPOUTL | R | 0h | CMPSS7_CTRIPOUTL X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 27 | CMPSS6_CTRIPOUTH | R | 0h | CMPSS6_CTRIPOUTH X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 26 | CMPSS6_CTRIPOUTL | R | 0h | CMPSS6_CTRIPOUTL X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 25 | CMPSS5_CTRIPOUTH | R | 0h | CMPSS5_CTRIPOUTH X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 24 | CMPSS5_CTRIPOUTL | R | 0h | CMPSS5_CTRIPOUTL X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 23 | CMPSS4_CTRIPOUTH | R | 0h | CMPSS4_CTRIPOUTH X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 22 | CMPSS4_CTRIPOUTL | R | 0h | CMPSS4_CTRIPOUTL X-BAR Flag<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-25. XBARFLG1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 21 | CMPSS3_CTRIPOUTH | R | 0h | CMPSS3_CTRIPOUTH X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 20 | CMPSS3_CTRIPOUTL | R | 0h | CMPSS3_CTRIPOUTL X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 19 | CMPSS2_CTRIPOUTH | R | 0h | CMPSS2_CTRIPOUTH X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 18 | CMPSS2_CTRIPOUTL | R | 0h | CMPSS2_CTRIPOUTL X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 17 | CMPSS1_CTRIPOUTH | R | 0h | CMPSS1_CTRIPOUTH X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 16 | CMPSS1_CTRIPOUTL | R | 0h | CMPSS1_CTRIPOUTL X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 15 | CMPSS8_CTRIPH | R | 0h | CMPSS8_CTRIPH X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 14 | CMPSS8_CTRIPL | R | 0h | CMPSS8_CTRIPL X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 13 | CMPSS7_CTRIPH | R | 0h | CMPSS7_CTRIPH X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 12 | CMPSS7_CTRIPL | R | 0h | CMPSS7_CTRIPL X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 11 | CMPSS6_CTRIPH | R | 0h | CMPSS6_CTRIPH X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 10 | CMPSS6_CTRIPL | R | 0h | CMPSS6_CTRIPL X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 9 | CMPSS5_CTRIPH | R | 0h | CMPSS5_CTRIPH X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 8 | CMPSS5_CTRIPL | R | 0h | CMPSS5_CTRIPL X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 7 | CMPSS4_CTRIPH | R | 0h | CMPSS4_CTRIPH X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 6 | CMPSS4_CTRIPL | R | 0h | CMPSS4_CTRIPL X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 5 | CMPSS3_CTRIPH | R | 0h | CMPSS3_CTRIPH X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 4 | CMPSS3_CTRIPL | R | 0h | CMPSS3_CTRIPL X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 3 | CMPSS2_CTRIPH | R | 0h | CMPSS2_CTRIPH X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 2 | CMPSS2_CTRIPL | R | 0h | CMPSS2_CTRIPL X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 1 | CMPSS1_CTRIPH | R | 0h | CMPSS1_CTRIPH X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 0 | CMPSS1_CTRIPL | R | 0h | CMPSS1_CTRIPL X-BAR Flag<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.3.2 XBARFLG2 Register (Offset = 2h) [Reset = 00000000h]

XBARFLG2 is shown in Figure 9-23 and described in Table 9-26.

Return to the Summary Table.

This register is used to flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.
1: Corresponding Input was triggered
0: Corresponding Input was not triggered

**Figure 9-23. XBARFLG2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| ADCCEVT1 | ADCBEVT4 | ADCBEVT3 | ADCBEVT2 | ADCBEVT1 | ADCAEVT4 | ADCAEVT3 | ADCAEVT2 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| ADCAEVT1 | EXTSYNCOUT | ECAP6_OUT | ECAP5_OUT | ECAP4_OUT | ECAP3_OUT | ECAP2_OUT | ECAP1_OUT |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| CLB4_OUT5 | CLB4_OUT4 | CLB3_OUT5 | CLB3_OUT4 | CLB2_OUT5 | CLB2_OUT4 | CLB1_OUT5 | CLB1_OUT4 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ADCSOCBO | ADCSOCAO | INPUT6 | INPUT5 | INPUT4 | INPUT3 | INPUT2 | INPUT1 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 9-26. XBARFLG2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | ADCCEVT1 | R | 0h | ADCCEVT1 X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 30 | ADCBEVT4 | R | 0h | ADCBEVT4 X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 29 | ADCBEVT3 | R | 0h | ADCBEVT3 X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 28 | ADCBEVT2 | R | 0h | ADCBEVT2 X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 27 | ADCBEVT1 | R | 0h | ADCBEVT1 X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 26 | ADCAEVT4 | R | 0h | ADCAEVT4 X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 25 | ADCAEVT3 | R | 0h | ADCAEVT3 X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 24 | ADCAEVT2 | R | 0h | ADCAEVT2 X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 23 | ADCAEVT1 | R | 0h | ADCAEVT1 X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 22 | EXTSYNCOUT | R | 0h | EXTSYNCOUT X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 21 | ECAP6_OUT | R | 0h | ECAP6_OUT X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 20 | ECAP5_OUT | R | 0h | ECAP5_OUT X-BAR Flag<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-26. XBARFLG2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19 | ECAP4_OUT | R | 0h | ECAP4_OUT X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 18 | ECAP3_OUT | R | 0h | ECAP3_OUT X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 17 | ECAP2_OUT | R | 0h | ECAP2_OUT X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 16 | ECAP1_OUT | R | 0h | ECAP1_OUT X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 15 | CLB4_OUT5 | R | 0h | CLB4_OUT5 X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 14 | CLB4_OUT4 | R | 0h | CLB4_OUT4 X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 13 | CLB3_OUT5 | R | 0h | CLB3_OUT5 X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 12 | CLB3_OUT4 | R | 0h | CLB3_OUT4 X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 11 | CLB2_OUT5 | R | 0h | CLB2_OUT5 X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 10 | CLB2_OUT4 | R | 0h | CLB2_OUT4 X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 9 | CLB1_OUT5 | R | 0h | CLB1_OUT5 X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 8 | CLB1_OUT4 | R | 0h | CLB1_OUT4 X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 7 | ADCSOCBO | R | 0h | ADCSOCBO X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 6 | ADCSOCAO | R | 0h | ADCSOCAO X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 5 | INPUT6 | R | 0h | INPUT6 X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 4 | INPUT5 | R | 0h | INPUT5 X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 3 | INPUT4 | R | 0h | INPUT4 X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 2 | INPUT3 | R | 0h | INPUT3 X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 1 | INPUT2 | R | 0h | INPUT2 X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 0 | INPUT1 | R | 0h | INPUT1 X-BAR Flag<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.3.3 XBARFLG3 Register (Offset = 4h) [Reset = 00000000h]

XBARFLG3 is shown in Figure 9-24 and described in Table 9-27.

Return to the Summary Table.

This register is used to flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.
1: Corresponding Input was triggered
0: Corresponding Input was not triggered

**Figure 9-24. XBARFLG3 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | SD2FLT4_COMPH | SD2FLT4_COMPL | SD2FLT3_COMPH | SD2FLT3_COMPL | SD2FLT2_COMPH | SD2FLT2_COMPL | SD2FLT1_COMPH |
| R-0-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| SD2FLT1_COMPL | SD1FLT4_COMPH | SD1FLT4_COMPL | SD1FLT3_COMPH | SD1FLT3_COMPL | SD1FLT2_COMPH | SD1FLT2_COMPL | SD1FLT1_COMPH |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SD1FLT1_COMPL | ADCDEVT4 | ADCDEVT3 | ADCDEVT2 | ADCDEVT1 | ADCCEVT4 | ADCCEVT3 | ADCCEVT2 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 9-27. XBARFLG3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-23 | RESERVED | R-0 | 0h | Reserved |
| 22 | SD2FLT4_COMPH | R | 0h | SD2FLT4_COMPH X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 21 | SD2FLT4_COMPL | R | 0h | SD2FLT4_COMPL X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 20 | SD2FLT3_COMPH | R | 0h | SD2FLT3_COMPH X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 19 | SD2FLT3_COMPL | R | 0h | SD2FLT3_COMPL X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 18 | SD2FLT2_COMPH | R | 0h | SD2FLT2_COMPH X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 17 | SD2FLT2_COMPL | R | 0h | SD2FLT2_COMPL X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 16 | SD2FLT1_COMPH | R | 0h | SD2FLT1_COMPH X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 15 | SD2FLT1_COMPL | R | 0h | SD2FLT1_COMPL X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 14 | SD1FLT4_COMPH | R | 0h | SD1FLT4_COMPH X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 13 | SD1FLT4_COMPL | R | 0h | SD1FLT4_COMPL X-BAR Flag<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-27. XBARFLG3 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 12 | SD1FLT3_COMPH | R | 0h | SD1FLT3_COMPH X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 11 | SD1FLT3_COMPL | R | 0h | SD1FLT3_COMPL X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 10 | SD1FLT2_COMPH | R | 0h | SD1FLT2_COMPH X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 9 | SD1FLT2_COMPL | R | 0h | SD1FLT2_COMPL X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 8 | SD1FLT1_COMPH | R | 0h | SD1FLT1_COMPH X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 7 | SD1FLT1_COMPL | R | 0h | SD1FLT1_COMPL X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 6 | ADCDEVT4 | R | 0h | ADCDEVT4 X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 5 | ADCDEVT3 | R | 0h | ADCDEVT3 X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 4 | ADCDEVT2 | R | 0h | ADCDEVT2 X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 3 | ADCDEVT1 | R | 0h | ADCDEVT1 X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 2 | ADCCEVT4 | R | 0h | ADCCEVT4 X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 1 | ADCCEVT3 | R | 0h | ADCCEVT3 X-BAR Flag<br>Reset type: CPU1.SYSRSn |
| 0 | ADCCEVT2 | R | 0h | ADCCEVT2 X-BAR Flag<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.3.4 XBARCLR1 Register (Offset = 8h) [Reset = 00000000h]

XBARCLR1 is shown in Figure 9-25 and described in Table 9-28.

Return to the Summary Table.

This register is used to clear the flag(s) in the XBARFLG1 register.
1: Clears the corresponding bit in the XBARFLG1 register.
0: Writing 0 has no effect

**Figure 9-25. XBARCLR1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| CMPSS8_CTRIPOUTH | CMPSS8_CTRIPOUTL | CMPSS7_CTRIPOUTH | CMPSS7_CTRIPOUTL | CMPSS6_CTRIPOUTH | CMPSS6_CTRIPOUTL | CMPSS5_CTRIPOUTH | CMPSS5_CTRIPOUTL |
| R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| CMPSS4_CTRIPOUTH | CMPSS4_CTRIPOUTL | CMPSS3_CTRIPOUTH | CMPSS3_CTRIPOUTL | CMPSS2_CTRIPOUTH | CMPSS2_CTRIPOUTL | CMPSS1_CTRIPOUTH | CMPSS1_CTRIPOUTL |
| R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| CMPSS8_CTRIPH | CMPSS8_CTRIPL | CMPSS7_CTRIPH | CMPSS7_CTRIPL | CMPSS6_CTRIPH | CMPSS6_CTRIPL | CMPSS5_CTRIPH | CMPSS5_CTRIPL |
| R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CMPSS4_CTRIPH | CMPSS4_CTRIPL | CMPSS3_CTRIPH | CMPSS3_CTRIPL | CMPSS2_CTRIPH | CMPSS2_CTRIPL | CMPSS1_CTRIPH | CMPSS1_CTRIPL |
| R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h |

**Table 9-28. XBARCLR1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | CMPSS8_CTRIPOUTH | R-0/W1S | 0h | CMPSS8_CTRIPOUTH X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 30 | CMPSS8_CTRIPOUTL | R-0/W1S | 0h | CMPSS8_CTRIPOUTL X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 29 | CMPSS7_CTRIPOUTH | R-0/W1S | 0h | CMPSS7_CTRIPOUTH X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 28 | CMPSS7_CTRIPOUTL | R-0/W1S | 0h | CMPSS7_CTRIPOUTL X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 27 | CMPSS6_CTRIPOUTH | R-0/W1S | 0h | CMPSS6_CTRIPOUTH X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 26 | CMPSS6_CTRIPOUTL | R-0/W1S | 0h | CMPSS6_CTRIPOUTL X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 25 | CMPSS5_CTRIPOUTH | R-0/W1S | 0h | CMPSS5_CTRIPOUTH X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 24 | CMPSS5_CTRIPOUTL | R-0/W1S | 0h | CMPSS5_CTRIPOUTL X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 23 | CMPSS4_CTRIPOUTH | R-0/W1S | 0h | CMPSS4_CTRIPOUTH X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 22 | CMPSS4_CTRIPOUTL | R-0/W1S | 0h | CMPSS4_CTRIPOUTL X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 21 | CMPSS3_CTRIPOUTH | R-0/W1S | 0h | CMPSS3_CTRIPOUTH X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-28. XBARCLR1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 20 | CMPSS3_CTRIPOUTL | R-0/W1S | 0h | CMPSS3_CTRIPOUTL X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 19 | CMPSS2_CTRIPOUTH | R-0/W1S | 0h | CMPSS2_CTRIPOUTH X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 18 | CMPSS2_CTRIPOUTL | R-0/W1S | 0h | CMPSS2_CTRIPOUTL X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 17 | CMPSS1_CTRIPOUTH | R-0/W1S | 0h | CMPSS1_CTRIPOUTH X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 16 | CMPSS1_CTRIPOUTL | R-0/W1S | 0h | CMPSS1_CTRIPOUTL X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 15 | CMPSS8_CTRIPH | R-0/W1S | 0h | CMPSS8_CTRIPH X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 14 | CMPSS8_CTRIPL | R-0/W1S | 0h | CMPSS8_CTRIPL X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 13 | CMPSS7_CTRIPH | R-0/W1S | 0h | CMPSS7_CTRIPH X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 12 | CMPSS7_CTRIPL | R-0/W1S | 0h | CMPSS7_CTRIPL X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 11 | CMPSS6_CTRIPH | R-0/W1S | 0h | CMPSS6_CTRIPH X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 10 | CMPSS6_CTRIPL | R-0/W1S | 0h | CMPSS6_CTRIPL X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 9 | CMPSS5_CTRIPH | R-0/W1S | 0h | CMPSS5_CTRIPH X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 8 | CMPSS5_CTRIPL | R-0/W1S | 0h | CMPSS5_CTRIPL X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 7 | CMPSS4_CTRIPH | R-0/W1S | 0h | CMPSS4_CTRIPH X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 6 | CMPSS4_CTRIPL | R-0/W1S | 0h | CMPSS4_CTRIPL X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 5 | CMPSS3_CTRIPH | R-0/W1S | 0h | CMPSS3_CTRIPH X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 4 | CMPSS3_CTRIPL | R-0/W1S | 0h | CMPSS3_CTRIPL X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 3 | CMPSS2_CTRIPH | R-0/W1S | 0h | CMPSS2_CTRIPH X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 2 | CMPSS2_CTRIPL | R-0/W1S | 0h | CMPSS2_CTRIPL X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 1 | CMPSS1_CTRIPH | R-0/W1S | 0h | CMPSS1_CTRIPH X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 0 | CMPSS1_CTRIPL | R-0/W1S | 0h | CMPSS1_CTRIPL X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.3.5 XBARCLR2 Register (Offset = Ah) [Reset = 00000000h]

XBARCLR2 is shown in Figure 9-26 and described in Table 9-29.

Return to the Summary Table.

This register is used to clear the flag(s) in the XBARFLG2 register.
1: Clears the corresponding bit in the XBARFLG2 register.
0: Writing 0 has no effect

**Figure 9-26. XBARCLR2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| ADCCEVT1 | ADCBEVT4 | ADCBEVT3 | ADCBEVT2 | ADCBEVT1 | ADCAEVT4 | ADCAEVT3 | ADCAEVT2 |
| R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| ADCAEVT1 | EXTSYNCOUT | ECAP6_OUT | ECAP5_OUT | ECAP4_OUT | ECAP3_OUT | ECAP2_OUT | ECAP1_OUT |
| R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| CLB4_OUT5 | CLB4_OUT4 | CLB3_OUT5 | CLB3_OUT4 | CLB2_OUT5 | CLB2_OUT4 | CLB1_OUT5 | CLB1_OUT4 |
| R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ADCSOCBO | ADCSOCAO | INPUT6 | INPUT5 | INPUT4 | INPUT3 | INPUT2 | INPUT1 |
| R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h |

**Table 9-29. XBARCLR2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | ADCCEVT1 | R-0/W1S | 0h | ADCCEVT1 X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 30 | ADCBEVT4 | R-0/W1S | 0h | ADCBEVT4 X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 29 | ADCBEVT3 | R-0/W1S | 0h | ADCBEVT3 X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 28 | ADCBEVT2 | R-0/W1S | 0h | ADCBEVT2 X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 27 | ADCBEVT1 | R-0/W1S | 0h | ADCBEVT1 X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 26 | ADCAEVT4 | R-0/W1S | 0h | ADCAEVT4 X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 25 | ADCAEVT3 | R-0/W1S | 0h | ADCAEVT3 X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 24 | ADCAEVT2 | R-0/W1S | 0h | ADCAEVT2 X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 23 | ADCAEVT1 | R-0/W1S | 0h | ADCAEVT1 X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 22 | EXTSYNCOUT | R-0/W1S | 0h | EXTSYNCOUT X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 21 | ECAP6_OUT | R-0/W1S | 0h | ECAP6_OUT X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 20 | ECAP5_OUT | R-0/W1S | 0h | ECAP5_OUT X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-29. XBARCLR2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19 | ECAP4_OUT | R-0/W1S | 0h | ECAP4_OUT X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 18 | ECAP3_OUT | R-0/W1S | 0h | ECAP3_OUT X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 17 | ECAP2_OUT | R-0/W1S | 0h | ECAP2_OUT X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 16 | ECAP1_OUT | R-0/W1S | 0h | ECAP1_OUT X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 15 | CLB4_OUT5 | R-0/W1S | 0h | CLB4_OUT5 X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 14 | CLB4_OUT4 | R-0/W1S | 0h | CLB4_OUT4 X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 13 | CLB3_OUT5 | R-0/W1S | 0h | CLB3_OUT5 X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 12 | CLB3_OUT4 | R-0/W1S | 0h | CLB3_OUT4 X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 11 | CLB2_OUT5 | R-0/W1S | 0h | CLB2_OUT5 X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 10 | CLB2_OUT4 | R-0/W1S | 0h | CLB2_OUT4 X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 9 | CLB1_OUT5 | R-0/W1S | 0h | CLB1_OUT5 X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 8 | CLB1_OUT4 | R-0/W1S | 0h | CLB1_OUT4 X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 7 | ADCSOCBO | R-0/W1S | 0h | ADCSOCBO X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 6 | ADCSOCAO | R-0/W1S | 0h | ADCSOCAO X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 5 | INPUT6 | R-0/W1S | 0h | INPUT6 X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 4 | INPUT5 | R-0/W1S | 0h | INPUT5 X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 3 | INPUT4 | R-0/W1S | 0h | INPUT4 X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 2 | INPUT3 | R-0/W1S | 0h | INPUT3 X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 1 | INPUT2 | R-0/W1S | 0h | INPUT2 X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 0 | INPUT1 | R-0/W1S | 0h | INPUT1 X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.3.6 XBARCLR3 Register (Offset = Ch) [Reset = 00000000h]

XBARCLR3 is shown in Figure 9-27 and described in Table 9-30.

Return to the Summary Table.

This register is used to clear the flag(s) in the XBARFLG3 register.
1: Clears the corresponding bit in the XBARFLG3 register.
0: Writing 0 has no effect

**Figure 9-27. XBARCLR3 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | SD2FLT4_COMPH | SD2FLT4_COMPL | SD2FLT3_COMPH | SD2FLT3_COMPL | SD2FLT2_COMPH | SD2FLT2_COMPL | SD2FLT1_COMPH |
| R-0-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| SD2FLT1_COMPL | SD1FLT4_COMPH | SD1FLT4_COMPL | SD1FLT3_COMPH | SD1FLT3_COMPL | SD1FLT2_COMPH | SD1FLT2_COMPL | SD1FLT1_COMPH |
| R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SD1FLT1_COMPL | ADCDEVT4 | ADCDEVT3 | ADCDEVT2 | ADCDEVT1 | ADCCEVT4 | ADCCEVT3 | ADCCEVT2 |
| R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h |

**Table 9-30. XBARCLR3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-23 | RESERVED | R-0 | 0h | Reserved |
| 22 | SD2FLT4_COMPH | R-0/W1S | 0h | SD2FLT4_COMPH X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 21 | SD2FLT4_COMPL | R-0/W1S | 0h | SD2FLT4_COMPL X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 20 | SD2FLT3_COMPH | R-0/W1S | 0h | SD2FLT3_COMPH X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 19 | SD2FLT3_COMPL | R-0/W1S | 0h | SD2FLT3_COMPL X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 18 | SD2FLT2_COMPH | R-0/W1S | 0h | SD2FLT2_COMPH X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 17 | SD2FLT2_COMPL | R-0/W1S | 0h | SD2FLT2_COMPL X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 16 | SD2FLT1_COMPH | R-0/W1S | 0h | SD2FLT1_COMPH X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 15 | SD2FLT1_COMPL | R-0/W1S | 0h | SD2FLT1_COMPL X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 14 | SD1FLT4_COMPH | R-0/W1S | 0h | SD1FLT4_COMPH X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 13 | SD1FLT4_COMPL | R-0/W1S | 0h | SD1FLT4_COMPL X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 12 | SD1FLT3_COMPH | R-0/W1S | 0h | SD1FLT3_COMPH X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-30. XBARCLR3 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 11 | SD1FLT3_COMPL | R-0/W1S | 0h | SD1FLT3_COMPL X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 10 | SD1FLT2_COMPH | R-0/W1S | 0h | SD1FLT2_COMPH X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 9 | SD1FLT2_COMPL | R-0/W1S | 0h | SD1FLT2_COMPL X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 8 | SD1FLT1_COMPH | R-0/W1S | 0h | SD1FLT1_COMPH X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 7 | SD1FLT1_COMPL | R-0/W1S | 0h | SD1FLT1_COMPL X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 6 | ADCDEVT4 | R-0/W1S | 0h | ADCDEVT4 X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 5 | ADCDEVT3 | R-0/W1S | 0h | ADCDEVT3 X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 4 | ADCDEVT2 | R-0/W1S | 0h | ADCDEVT2 X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 3 | ADCDEVT1 | R-0/W1S | 0h | ADCDEVT1 X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 2 | ADCCEVT4 | R-0/W1S | 0h | ADCCEVT4 X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 1 | ADCCEVT3 | R-0/W1S | 0h | ADCCEVT3 X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |
| 0 | ADCCEVT2 | R-0/W1S | 0h | ADCCEVT2 X-BAR Flag Clear<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.4 EPWM_XBAR_REGS Registers

Table 9-31 lists the memory-mapped registers for the EPWM_XBAR_REGS registers. All register offset addresses not listed in Table 9-31 should be considered as reserved locations and the register contents should not be modified.

**Table 9-31. EPWM_XBAR_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | TRIP4MUX0TO15CFG | ePWM XBAR Mux Configuration for TRIP4 | EALLOW | Go |
| 2h | TRIP4MUX16TO31CFG | ePWM XBAR Mux Configuration for TRIP4 | EALLOW | Go |
| 4h | TRIP5MUX0TO15CFG | ePWM XBAR Mux Configuration for TRIP5 | EALLOW | Go |
| 6h | TRIP5MUX16TO31CFG | ePWM XBAR Mux Configuration for TRIP5 | EALLOW | Go |
| 8h | TRIP7MUX0TO15CFG | ePWM XBAR Mux Configuration for TRIP7 | EALLOW | Go |
| Ah | TRIP7MUX16TO31CFG | ePWM XBAR Mux Configuration for TRIP7 | EALLOW | Go |
| Ch | TRIP8MUX0TO15CFG | ePWM XBAR Mux Configuration for TRIP8 | EALLOW | Go |
| Eh | TRIP8MUX16TO31CFG | ePWM XBAR Mux Configuration for TRIP8 | EALLOW | Go |
| 10h | TRIP9MUX0TO15CFG | ePWM XBAR Mux Configuration for TRIP9 | EALLOW | Go |
| 12h | TRIP9MUX16TO31CFG | ePWM XBAR Mux Configuration for TRIP9 | EALLOW | Go |
| 14h | TRIP10MUX0TO15CFG | ePWM XBAR Mux Configuration for TRIP10 | EALLOW | Go |
| 16h | TRIP10MUX16TO31CFG | ePWM XBAR Mux Configuration for TRIP10 | EALLOW | Go |
| 18h | TRIP11MUX0TO15CFG | ePWM XBAR Mux Configuration for TRIP11 | EALLOW | Go |
| 1Ah | TRIP11MUX16TO31CFG | ePWM XBAR Mux Configuration for TRIP11 | EALLOW | Go |
| 1Ch | TRIP12MUX0TO15CFG | ePWM XBAR Mux Configuration for TRIP12 | EALLOW | Go |
| 1Eh | TRIP12MUX16TO31CFG | ePWM XBAR Mux Configuration for TRIP12 | EALLOW | Go |
| 20h | TRIP4MUXENABLE | ePWM XBAR Mux Enable for TRIP4 | EALLOW | Go |
| 22h | TRIP5MUXENABLE | ePWM XBAR Mux Enable for TRIP5 | EALLOW | Go |
| 24h | TRIP7MUXENABLE | ePWM XBAR Mux Enable for TRIP7 | EALLOW | Go |
| 26h | TRIP8MUXENABLE | ePWM XBAR Mux Enable for TRIP8 | EALLOW | Go |
| 28h | TRIP9MUXENABLE | ePWM XBAR Mux Enable for TRIP9 | EALLOW | Go |
| 2Ah | TRIP10MUXENABLE | ePWM XBAR Mux Enable for TRIP10 | EALLOW | Go |
| 2Ch | TRIP11MUXENABLE | ePWM XBAR Mux Enable for TRIP11 | EALLOW | Go |
| 2Eh | TRIP12MUXENABLE | ePWM XBAR Mux Enable for TRIP12 | EALLOW | Go |
| 38h | TRIPOUTINV | ePWM XBAR Output Inversion Register | EALLOW | Go |
| 3Eh | TRIPLOCK | ePWM XBAR Configuration Lock register | EALLOW | Go |

Complex bit access types are encoded to fit into small table cells. Table 9-32 shows the codes that are used for access types in this section.

**Table 9-32. EPWM_XBAR_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| R-0 | R<br>-0 | Read<br>Returns 0s |
| Write Type | | |
| W | W | Write |
| WSonce | W<br>Sonce | Write<br>Set once |
| Reset or Default Value | | |
| *-n* | | Value after reset or the default value |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-32. 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. |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.4.1 TRIP4MUX0TO15CFG Register (Offset = 0h) [Reset = 00000000h]

TRIP4MUX0TO15CFG is shown in Figure 9-28 and described in Table 9-33.

Return to the Summary Table.

ePWM XBAR Mux Configuration for TRIP4

**Figure 9-28. 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 9-33. TRIP4MUX0TO15CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX15 | R/W | 0h | Select Bits for EPWM-XBAR TRIP4 Mux15:<br>00 : Select .0 input for Mux15<br>01 : Select .1 input for Mux15<br>10 : Select .2 input for Mux15<br>11 : Select .3 input for Mux15<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX14 | R/W | 0h | Select Bits for EPWM-XBAR TRIP4 Mux14:<br>00 : Select .0 input for Mux14<br>01 : Select .1 input for Mux14<br>10 : Select .2 input for Mux14<br>11 : Select .3 input for Mux14<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX13 | R/W | 0h | Select Bits for EPWM-XBAR TRIP4 Mux13:<br>00 : Select .0 input for Mux13<br>01 : Select .1 input for Mux13<br>10 : Select .2 input for Mux13<br>11 : Select .3 input for Mux13<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX12 | R/W | 0h | Select Bits for EPWM-XBAR TRIP4 Mux12:<br>00 : Select .0 input for Mux12<br>01 : Select .1 input for Mux12<br>10 : Select .2 input for Mux12<br>11 : Select .3 input for Mux12<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX11 | R/W | 0h | Select Bits for EPWM-XBAR TRIP4 Mux11:<br>00 : Select .0 input for Mux11<br>01 : Select .1 input for Mux11<br>10 : Select .2 input for Mux11<br>11 : Select .3 input for Mux11<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21-20 | MUX10 | R/W | 0h | Select Bits for EPWM-XBAR TRIP4 Mux10:<br>00 : Select .0 input for Mux10<br>01 : Select .1 input for Mux10<br>10 : Select .2 input for Mux10<br>11 : Select .3 input for Mux10<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-33. TRIP4MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19-18 | MUX9 | R/W | 0h | Select Bits for EPWM-XBAR TRIP4 Mux9:<br>00 : Select .0 input for Mux9<br>01 : Select .1 input for Mux9<br>10 : Select .2 input for Mux9<br>11 : Select .3 input for Mux9<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX8 | R/W | 0h | Select Bits for EPWM-XBAR TRIP4 Mux8:<br>00 : Select .0 input for Mux8<br>01 : Select .1 input for Mux8<br>10 : Select .2 input for Mux8<br>11 : Select .3 input for Mux8<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX7 | R/W | 0h | Select Bits for EPWM-XBAR TRIP4 Mux7:<br>00 : Select .0 input for Mux7<br>01 : Select .1 input for Mux7<br>10 : Select .2 input for Mux7<br>11 : Select .3 input for Mux7<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX6 | R/W | 0h | Select Bits for EPWM-XBAR TRIP4 Mux6:<br>00 : Select .0 input for Mux6<br>01 : Select .1 input for Mux6<br>10 : Select .2 input for Mux6<br>11 : Select .3 input for Mux6<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX5 | R/W | 0h | Select Bits for EPWM-XBAR TRIP4 Mux5:<br>00 : Select .0 input for Mux5<br>01 : Select .1 input for Mux5<br>10 : Select .2 input for Mux5<br>11 : Select .3 input for Mux5<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX4 | R/W | 0h | Select Bits for EPWM-XBAR TRIP4 Mux4:<br>00 : Select .0 input for Mux4<br>01 : Select .1 input for Mux4<br>10 : Select .2 input for Mux4<br>11 : Select .3 input for Mux4<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7-6 | MUX3 | R/W | 0h | Select Bits for EPWM-XBAR TRIP4 Mux3:<br>00 : Select .0 input for Mux3<br>01 : Select .1 input for Mux3<br>10 : Select .2 input for Mux3<br>11 : Select .3 input for Mux3<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX2 | R/W | 0h | Select Bits for EPWM-XBAR TRIP4 Mux2:<br>00 : Select .0 input for Mux2<br>01 : Select .1 input for Mux2<br>10 : Select .2 input for Mux2<br>11 : Select .3 input for Mux2<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-33. TRIP4MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-2 | MUX1 | R/W | 0h | Select Bits for EPWM-XBAR TRIP4 Mux1:<br>00 : Select .0 input for Mux1<br>01 : Select .1 input for Mux1<br>10 : Select .2 input for Mux1<br>11 : Select .3 input for Mux1<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX0 | R/W | 0h | Select Bits for EPWM-XBAR TRIP4 Mux0:<br>00 : Select .0 input for Mux0<br>01 : Select .1 input for Mux0<br>10 : Select .2 input for Mux0<br>11 : Select .3 input for Mux0<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.4.2 TRIP4MUX16TO31CFG Register (Offset = 2h) [Reset = 00000000h]

TRIP4MUX16TO31CFG is shown in Figure 9-29 and described in Table 9-34.

Return to the Summary Table.

ePWM XBAR Mux Configuration for TRIP4

**Figure 9-29. 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 9-34. TRIP4MUX16TO31CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX31 | R/W | 0h | Select Bits for EPWM-XBAR TRIP4 Mux31:<br>00 : Select .0 input for Mux31<br>01 : Select .1 input for Mux31<br>10 : Select .2 input for Mux31<br>11 : Select .3 input for Mux31<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX30 | R/W | 0h | Select Bits for EPWM-XBAR TRIP4 Mux30:<br>00 : Select .0 input for Mux30<br>01 : Select .1 input for Mux30<br>10 : Select .2 input for Mux30<br>11 : Select .3 input for Mux30<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX29 | R/W | 0h | Select Bits for EPWM-XBAR TRIP4 Mux29:<br>00 : Select .0 input for Mux29<br>01 : Select .1 input for Mux29<br>10 : Select .2 input for Mux29<br>11 : Select .3 input for Mux29<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX28 | R/W | 0h | Select Bits for EPWM-XBAR TRIP4 Mux28:<br>00 : Select .0 input for Mux28<br>01 : Select .1 input for Mux28<br>10 : Select .2 input for Mux28<br>11 : Select .3 input for Mux28<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX27 | R/W | 0h | Select Bits for EPWM-XBAR TRIP4 Mux27:<br>00 : Select .0 input for Mux27<br>01 : Select .1 input for Mux27<br>10 : Select .2 input for Mux27<br>11 : Select .3 input for Mux27<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21-20 | MUX26 | R/W | 0h | Select Bits for EPWM-XBAR TRIP4 Mux26:<br>00 : Select .0 input for Mux26<br>01 : Select .1 input for Mux26<br>10 : Select .2 input for Mux26<br>11 : Select .3 input for Mux26<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-34. TRIP4MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19-18 | MUX25 | R/W | 0h | Select Bits for EPWM-XBAR TRIP4 Mux25:<br>00 : Select .0 input for Mux25<br>01 : Select .1 input for Mux25<br>10 : Select .2 input for Mux25<br>11 : Select .3 input for Mux25<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX24 | R/W | 0h | Select Bits for EPWM-XBAR TRIP4 Mux24:<br>00 : Select .0 input for Mux24<br>01 : Select .1 input for Mux24<br>10 : Select .2 input for Mux24<br>11 : Select .3 input for Mux24<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX23 | R/W | 0h | Select Bits for EPWM-XBAR TRIP4 Mux23:<br>00 : Select .0 input for Mux23<br>01 : Select .1 input for Mux23<br>10 : Select .2 input for Mux23<br>11 : Select .3 input for Mux23<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX22 | R/W | 0h | Select Bits for EPWM-XBAR TRIP4 Mux22:<br>00 : Select .0 input for Mux22<br>01 : Select .1 input for Mux22<br>10 : Select .2 input for Mux22<br>11 : Select .3 input for Mux22<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX21 | R/W | 0h | Select Bits for EPWM-XBAR TRIP4 Mux21:<br>00 : Select .0 input for Mux21<br>01 : Select .1 input for Mux21<br>10 : Select .2 input for Mux21<br>11 : Select .3 input for Mux21<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX20 | R/W | 0h | Select Bits for EPWM-XBAR TRIP4 Mux20:<br>00 : Select .0 input for Mux20<br>01 : Select .1 input for Mux20<br>10 : Select .2 input for Mux20<br>11 : Select .3 input for Mux20<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7-6 | MUX19 | R/W | 0h | Select Bits for EPWM-XBAR TRIP4 Mux19:<br>00 : Select .0 input for Mux19<br>01 : Select .1 input for Mux19<br>10 : Select .2 input for Mux19<br>11 : Select .3 input for Mux19<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX18 | R/W | 0h | Select Bits for EPWM-XBAR TRIP4 Mux18:<br>00 : Select .0 input for Mux18<br>01 : Select .1 input for Mux18<br>10 : Select .2 input for Mux18<br>11 : Select .3 input for Mux18<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-34. TRIP4MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-2 | MUX17 | R/W | 0h | Select Bits for EPWM-XBAR TRIP4 Mux17:<br>00 : Select .0 input for Mux17<br>01 : Select .1 input for Mux17<br>10 : Select .2 input for Mux17<br>11 : Select .3 input for Mux17<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX16 | R/W | 0h | Select Bits for EPWM-XBAR TRIP4 Mux16:<br>00 : Select .0 input for Mux16<br>01 : Select .1 input for Mux16<br>10 : Select .2 input for Mux16<br>11 : Select .3 input for Mux16<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.4.3 TRIP5MUX0TO15CFG Register (Offset = 4h) [Reset = 00000000h]

TRIP5MUX0TO15CFG is shown in Figure 9-30 and described in Table 9-35.

Return to the Summary Table.

ePWM XBAR Mux Configuration for TRIP5

**Figure 9-30. 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 9-35. TRIP5MUX0TO15CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX15 | R/W | 0h | Select Bits for EPWM-XBAR TRIP5 Mux15:<br>00 : Select .0 input for Mux15<br>01 : Select .1 input for Mux15<br>10 : Select .2 input for Mux15<br>11 : Select .3 input for Mux15<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX14 | R/W | 0h | Select Bits for EPWM-XBAR TRIP5 Mux14:<br>00 : Select .0 input for Mux14<br>01 : Select .1 input for Mux14<br>10 : Select .2 input for Mux14<br>11 : Select .3 input for Mux14<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX13 | R/W | 0h | Select Bits for EPWM-XBAR TRIP5 Mux13:<br>00 : Select .0 input for Mux13<br>01 : Select .1 input for Mux13<br>10 : Select .2 input for Mux13<br>11 : Select .3 input for Mux13<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX12 | R/W | 0h | Select Bits for EPWM-XBAR TRIP5 Mux12:<br>00 : Select .0 input for Mux12<br>01 : Select .1 input for Mux12<br>10 : Select .2 input for Mux12<br>11 : Select .3 input for Mux12<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX11 | R/W | 0h | Select Bits for EPWM-XBAR TRIP5 Mux11:<br>00 : Select .0 input for Mux11<br>01 : Select .1 input for Mux11<br>10 : Select .2 input for Mux11<br>11 : Select .3 input for Mux11<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21-20 | MUX10 | R/W | 0h | Select Bits for EPWM-XBAR TRIP5 Mux10:<br>00 : Select .0 input for Mux10<br>01 : Select .1 input for Mux10<br>10 : Select .2 input for Mux10<br>11 : Select .3 input for Mux10<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-35. TRIP5MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19-18 | MUX9 | R/W | 0h | Select Bits for EPWM-XBAR TRIP5 Mux9:<br>00 : Select .0 input for Mux9<br>01 : Select .1 input for Mux9<br>10 : Select .2 input for Mux9<br>11 : Select .3 input for Mux9<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX8 | R/W | 0h | Select Bits for EPWM-XBAR TRIP5 Mux8:<br>00 : Select .0 input for Mux8<br>01 : Select .1 input for Mux8<br>10 : Select .2 input for Mux8<br>11 : Select .3 input for Mux8<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX7 | R/W | 0h | Select Bits for EPWM-XBAR TRIP5 Mux7:<br>00 : Select .0 input for Mux7<br>01 : Select .1 input for Mux7<br>10 : Select .2 input for Mux7<br>11 : Select .3 input for Mux7<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX6 | R/W | 0h | Select Bits for EPWM-XBAR TRIP5 Mux6:<br>00 : Select .0 input for Mux6<br>01 : Select .1 input for Mux6<br>10 : Select .2 input for Mux6<br>11 : Select .3 input for Mux6<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX5 | R/W | 0h | Select Bits for EPWM-XBAR TRIP5 Mux5:<br>00 : Select .0 input for Mux5<br>01 : Select .1 input for Mux5<br>10 : Select .2 input for Mux5<br>11 : Select .3 input for Mux5<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX4 | R/W | 0h | Select Bits for EPWM-XBAR TRIP5 Mux4:<br>00 : Select .0 input for Mux4<br>01 : Select .1 input for Mux4<br>10 : Select .2 input for Mux4<br>11 : Select .3 input for Mux4<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7-6 | MUX3 | R/W | 0h | Select Bits for EPWM-XBAR TRIP5 Mux3:<br>00 : Select .0 input for Mux3<br>01 : Select .1 input for Mux3<br>10 : Select .2 input for Mux3<br>11 : Select .3 input for Mux3<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX2 | R/W | 0h | Select Bits for EPWM-XBAR TRIP5 Mux2:<br>00 : Select .0 input for Mux2<br>01 : Select .1 input for Mux2<br>10 : Select .2 input for Mux2<br>11 : Select .3 input for Mux2<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-35. TRIP5MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-2 | MUX1 | R/W | 0h | Select Bits for EPWM-XBAR TRIP5 Mux1:<br>00 : Select .0 input for Mux1<br>01 : Select .1 input for Mux1<br>10 : Select .2 input for Mux1<br>11 : Select .3 input for Mux1<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX0 | R/W | 0h | Select Bits for EPWM-XBAR TRIP5 Mux0:<br>00 : Select .0 input for Mux0<br>01 : Select .1 input for Mux0<br>10 : Select .2 input for Mux0<br>11 : Select .3 input for Mux0<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.4.4 TRIP5MUX16TO31CFG Register (Offset = 6h) [Reset = 00000000h]

TRIP5MUX16TO31CFG is shown in Figure 9-31 and described in Table 9-36.

Return to the Summary Table.

ePWM XBAR Mux Configuration for TRIP5

**Figure 9-31. 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 9-36. TRIP5MUX16TO31CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX31 | R/W | 0h | Select Bits for EPWM-XBAR TRIP5 Mux31:<br>00 : Select .0 input for Mux31<br>01 : Select .1 input for Mux31<br>10 : Select .2 input for Mux31<br>11 : Select .3 input for Mux31<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX30 | R/W | 0h | Select Bits for EPWM-XBAR TRIP5 Mux30:<br>00 : Select .0 input for Mux30<br>01 : Select .1 input for Mux30<br>10 : Select .2 input for Mux30<br>11 : Select .3 input for Mux30<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX29 | R/W | 0h | Select Bits for EPWM-XBAR TRIP5 Mux29:<br>00 : Select .0 input for Mux29<br>01 : Select .1 input for Mux29<br>10 : Select .2 input for Mux29<br>11 : Select .3 input for Mux29<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX28 | R/W | 0h | Select Bits for EPWM-XBAR TRIP5 Mux28:<br>00 : Select .0 input for Mux28<br>01 : Select .1 input for Mux28<br>10 : Select .2 input for Mux28<br>11 : Select .3 input for Mux28<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX27 | R/W | 0h | Select Bits for EPWM-XBAR TRIP5 Mux27:<br>00 : Select .0 input for Mux27<br>01 : Select .1 input for Mux27<br>10 : Select .2 input for Mux27<br>11 : Select .3 input for Mux27<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21-20 | MUX26 | R/W | 0h | Select Bits for EPWM-XBAR TRIP5 Mux26:<br>00 : Select .0 input for Mux26<br>01 : Select .1 input for Mux26<br>10 : Select .2 input for Mux26<br>11 : Select .3 input for Mux26<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-36. TRIP5MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19-18 | MUX25 | R/W | 0h | Select Bits for EPWM-XBAR TRIP5 Mux25:<br>00 : Select .0 input for Mux25<br>01 : Select .1 input for Mux25<br>10 : Select .2 input for Mux25<br>11 : Select .3 input for Mux25<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX24 | R/W | 0h | Select Bits for EPWM-XBAR TRIP5 Mux24:<br>00 : Select .0 input for Mux24<br>01 : Select .1 input for Mux24<br>10 : Select .2 input for Mux24<br>11 : Select .3 input for Mux24<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX23 | R/W | 0h | Select Bits for EPWM-XBAR TRIP5 Mux23:<br>00 : Select .0 input for Mux23<br>01 : Select .1 input for Mux23<br>10 : Select .2 input for Mux23<br>11 : Select .3 input for Mux23<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX22 | R/W | 0h | Select Bits for EPWM-XBAR TRIP5 Mux22:<br>00 : Select .0 input for Mux22<br>01 : Select .1 input for Mux22<br>10 : Select .2 input for Mux22<br>11 : Select .3 input for Mux22<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX21 | R/W | 0h | Select Bits for EPWM-XBAR TRIP5 Mux21:<br>00 : Select .0 input for Mux21<br>01 : Select .1 input for Mux21<br>10 : Select .2 input for Mux21<br>11 : Select .3 input for Mux21<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX20 | R/W | 0h | Select Bits for EPWM-XBAR TRIP5 Mux20:<br>00 : Select .0 input for Mux20<br>01 : Select .1 input for Mux20<br>10 : Select .2 input for Mux20<br>11 : Select .3 input for Mux20<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7-6 | MUX19 | R/W | 0h | Select Bits for EPWM-XBAR TRIP5 Mux19:<br>00 : Select .0 input for Mux19<br>01 : Select .1 input for Mux19<br>10 : Select .2 input for Mux19<br>11 : Select .3 input for Mux19<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX18 | R/W | 0h | Select Bits for EPWM-XBAR TRIP5 Mux18:<br>00 : Select .0 input for Mux18<br>01 : Select .1 input for Mux18<br>10 : Select .2 input for Mux18<br>11 : Select .3 input for Mux18<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-36. TRIP5MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-2 | MUX17 | R/W | 0h | Select Bits for EPWM-XBAR TRIP5 Mux17:<br>00 : Select .0 input for Mux17<br>01 : Select .1 input for Mux17<br>10 : Select .2 input for Mux17<br>11 : Select .3 input for Mux17<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX16 | R/W | 0h | Select Bits for EPWM-XBAR TRIP5 Mux16:<br>00 : Select .0 input for Mux16<br>01 : Select .1 input for Mux16<br>10 : Select .2 input for Mux16<br>11 : Select .3 input for Mux16<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.4.5 TRIP7MUX0TO15CFG Register (Offset = 8h) [Reset = 00000000h]

TRIP7MUX0TO15CFG is shown in Figure 9-32 and described in Table 9-37.

Return to the Summary Table.

ePWM XBAR Mux Configuration for TRIP7

**Figure 9-32. 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 9-37. TRIP7MUX0TO15CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX15 | R/W | 0h | Select Bits for EPWM-XBAR TRIP7 Mux15:<br>00 : Select .0 input for Mux15<br>01 : Select .1 input for Mux15<br>10 : Select .2 input for Mux15<br>11 : Select .3 input for Mux15<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX14 | R/W | 0h | Select Bits for EPWM-XBAR TRIP7 Mux14:<br>00 : Select .0 input for Mux14<br>01 : Select .1 input for Mux14<br>10 : Select .2 input for Mux14<br>11 : Select .3 input for Mux14<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX13 | R/W | 0h | Select Bits for EPWM-XBAR TRIP7 Mux13:<br>00 : Select .0 input for Mux13<br>01 : Select .1 input for Mux13<br>10 : Select .2 input for Mux13<br>11 : Select .3 input for Mux13<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX12 | R/W | 0h | Select Bits for EPWM-XBAR TRIP7 Mux12:<br>00 : Select .0 input for Mux12<br>01 : Select .1 input for Mux12<br>10 : Select .2 input for Mux12<br>11 : Select .3 input for Mux12<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX11 | R/W | 0h | Select Bits for EPWM-XBAR TRIP7 Mux11:<br>00 : Select .0 input for Mux11<br>01 : Select .1 input for Mux11<br>10 : Select .2 input for Mux11<br>11 : Select .3 input for Mux11<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21-20 | MUX10 | R/W | 0h | Select Bits for EPWM-XBAR TRIP7 Mux10:<br>00 : Select .0 input for Mux10<br>01 : Select .1 input for Mux10<br>10 : Select .2 input for Mux10<br>11 : Select .3 input for Mux10<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-37. TRIP7MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19-18 | MUX9 | R/W | 0h | Select Bits for EPWM-XBAR TRIP7 Mux9:<br>00 : Select .0 input for Mux9<br>01 : Select .1 input for Mux9<br>10 : Select .2 input for Mux9<br>11 : Select .3 input for Mux9<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX8 | R/W | 0h | Select Bits for EPWM-XBAR TRIP7 Mux8:<br>00 : Select .0 input for Mux8<br>01 : Select .1 input for Mux8<br>10 : Select .2 input for Mux8<br>11 : Select .3 input for Mux8<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX7 | R/W | 0h | Select Bits for EPWM-XBAR TRIP7 Mux7:<br>00 : Select .0 input for Mux7<br>01 : Select .1 input for Mux7<br>10 : Select .2 input for Mux7<br>11 : Select .3 input for Mux7<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX6 | R/W | 0h | Select Bits for EPWM-XBAR TRIP7 Mux6:<br>00 : Select .0 input for Mux6<br>01 : Select .1 input for Mux6<br>10 : Select .2 input for Mux6<br>11 : Select .3 input for Mux6<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX5 | R/W | 0h | Select Bits for EPWM-XBAR TRIP7 Mux5:<br>00 : Select .0 input for Mux5<br>01 : Select .1 input for Mux5<br>10 : Select .2 input for Mux5<br>11 : Select .3 input for Mux5<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX4 | R/W | 0h | Select Bits for EPWM-XBAR TRIP7 Mux4:<br>00 : Select .0 input for Mux4<br>01 : Select .1 input for Mux4<br>10 : Select .2 input for Mux4<br>11 : Select .3 input for Mux4<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7-6 | MUX3 | R/W | 0h | Select Bits for EPWM-XBAR TRIP7 Mux3:<br>00 : Select .0 input for Mux3<br>01 : Select .1 input for Mux3<br>10 : Select .2 input for Mux3<br>11 : Select .3 input for Mux3<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX2 | R/W | 0h | Select Bits for EPWM-XBAR TRIP7 Mux2:<br>00 : Select .0 input for Mux2<br>01 : Select .1 input for Mux2<br>10 : Select .2 input for Mux2<br>11 : Select .3 input for Mux2<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-37. TRIP7MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-2 | MUX1 | R/W | 0h | Select Bits for EPWM-XBAR TRIP7 Mux1:<br>00 : Select .0 input for Mux1<br>01 : Select .1 input for Mux1<br>10 : Select .2 input for Mux1<br>11 : Select .3 input for Mux1<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX0 | R/W | 0h | Select Bits for EPWM-XBAR TRIP7 Mux0:<br>00 : Select .0 input for Mux0<br>01 : Select .1 input for Mux0<br>10 : Select .2 input for Mux0<br>11 : Select .3 input for Mux0<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.4.6 TRIP7MUX16TO31CFG Register (Offset = Ah) [Reset = 00000000h]

TRIP7MUX16TO31CFG is shown in Figure 9-33 and described in Table 9-38.

Return to the Summary Table.

ePWM XBAR Mux Configuration for TRIP7

**Figure 9-33. 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 9-38. TRIP7MUX16TO31CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX31 | R/W | 0h | Select Bits for EPWM-XBAR TRIP7 Mux31:<br>00 : Select .0 input for Mux31<br>01 : Select .1 input for Mux31<br>10 : Select .2 input for Mux31<br>11 : Select .3 input for Mux31<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX30 | R/W | 0h | Select Bits for EPWM-XBAR TRIP7 Mux30:<br>00 : Select .0 input for Mux30<br>01 : Select .1 input for Mux30<br>10 : Select .2 input for Mux30<br>11 : Select .3 input for Mux30<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX29 | R/W | 0h | Select Bits for EPWM-XBAR TRIP7 Mux29:<br>00 : Select .0 input for Mux29<br>01 : Select .1 input for Mux29<br>10 : Select .2 input for Mux29<br>11 : Select .3 input for Mux29<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX28 | R/W | 0h | Select Bits for EPWM-XBAR TRIP7 Mux28:<br>00 : Select .0 input for Mux28<br>01 : Select .1 input for Mux28<br>10 : Select .2 input for Mux28<br>11 : Select .3 input for Mux28<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX27 | R/W | 0h | Select Bits for EPWM-XBAR TRIP7 Mux27:<br>00 : Select .0 input for Mux27<br>01 : Select .1 input for Mux27<br>10 : Select .2 input for Mux27<br>11 : Select .3 input for Mux27<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21-20 | MUX26 | R/W | 0h | Select Bits for EPWM-XBAR TRIP7 Mux26:<br>00 : Select .0 input for Mux26<br>01 : Select .1 input for Mux26<br>10 : Select .2 input for Mux26<br>11 : Select .3 input for Mux26<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-38. TRIP7MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19-18 | MUX25 | R/W | 0h | Select Bits for EPWM-XBAR TRIP7 Mux25:<br>00 : Select .0 input for Mux25<br>01 : Select .1 input for Mux25<br>10 : Select .2 input for Mux25<br>11 : Select .3 input for Mux25<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX24 | R/W | 0h | Select Bits for EPWM-XBAR TRIP7 Mux24:<br>00 : Select .0 input for Mux24<br>01 : Select .1 input for Mux24<br>10 : Select .2 input for Mux24<br>11 : Select .3 input for Mux24<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX23 | R/W | 0h | Select Bits for EPWM-XBAR TRIP7 Mux23:<br>00 : Select .0 input for Mux23<br>01 : Select .1 input for Mux23<br>10 : Select .2 input for Mux23<br>11 : Select .3 input for Mux23<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX22 | R/W | 0h | Select Bits for EPWM-XBAR TRIP7 Mux22:<br>00 : Select .0 input for Mux22<br>01 : Select .1 input for Mux22<br>10 : Select .2 input for Mux22<br>11 : Select .3 input for Mux22<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX21 | R/W | 0h | Select Bits for EPWM-XBAR TRIP7 Mux21:<br>00 : Select .0 input for Mux21<br>01 : Select .1 input for Mux21<br>10 : Select .2 input for Mux21<br>11 : Select .3 input for Mux21<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX20 | R/W | 0h | Select Bits for EPWM-XBAR TRIP7 Mux20:<br>00 : Select .0 input for Mux20<br>01 : Select .1 input for Mux20<br>10 : Select .2 input for Mux20<br>11 : Select .3 input for Mux20<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7-6 | MUX19 | R/W | 0h | Select Bits for EPWM-XBAR TRIP7 Mux19:<br>00 : Select .0 input for Mux19<br>01 : Select .1 input for Mux19<br>10 : Select .2 input for Mux19<br>11 : Select .3 input for Mux19<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX18 | R/W | 0h | Select Bits for EPWM-XBAR TRIP7 Mux18:<br>00 : Select .0 input for Mux18<br>01 : Select .1 input for Mux18<br>10 : Select .2 input for Mux18<br>11 : Select .3 input for Mux18<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-38. TRIP7MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-2 | MUX17 | R/W | 0h | Select Bits for EPWM-XBAR TRIP7 Mux17:<br>00 : Select .0 input for Mux17<br>01 : Select .1 input for Mux17<br>10 : Select .2 input for Mux17<br>11 : Select .3 input for Mux17<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX16 | R/W | 0h | Select Bits for EPWM-XBAR TRIP7 Mux16:<br>00 : Select .0 input for Mux16<br>01 : Select .1 input for Mux16<br>10 : Select .2 input for Mux16<br>11 : Select .3 input for Mux16<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.4.7 TRIP8MUX0TO15CFG Register (Offset = Ch) [Reset = 00000000h]

TRIP8MUX0TO15CFG is shown in Figure 9-34 and described in Table 9-39.

Return to the Summary Table.

ePWM XBAR Mux Configuration for TRIP8

**Figure 9-34. 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 9-39. TRIP8MUX0TO15CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX15 | R/W | 0h | Select Bits for EPWM-XBAR TRIP8 Mux15:<br>00 : Select .0 input for Mux15<br>01 : Select .1 input for Mux15<br>10 : Select .2 input for Mux15<br>11 : Select .3 input for Mux15<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX14 | R/W | 0h | Select Bits for EPWM-XBAR TRIP8 Mux14:<br>00 : Select .0 input for Mux14<br>01 : Select .1 input for Mux14<br>10 : Select .2 input for Mux14<br>11 : Select .3 input for Mux14<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX13 | R/W | 0h | Select Bits for EPWM-XBAR TRIP8 Mux13:<br>00 : Select .0 input for Mux13<br>01 : Select .1 input for Mux13<br>10 : Select .2 input for Mux13<br>11 : Select .3 input for Mux13<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX12 | R/W | 0h | Select Bits for EPWM-XBAR TRIP8 Mux12:<br>00 : Select .0 input for Mux12<br>01 : Select .1 input for Mux12<br>10 : Select .2 input for Mux12<br>11 : Select .3 input for Mux12<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX11 | R/W | 0h | Select Bits for EPWM-XBAR TRIP8 Mux11:<br>00 : Select .0 input for Mux11<br>01 : Select .1 input for Mux11<br>10 : Select .2 input for Mux11<br>11 : Select .3 input for Mux11<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21-20 | MUX10 | R/W | 0h | Select Bits for EPWM-XBAR TRIP8 Mux10:<br>00 : Select .0 input for Mux10<br>01 : Select .1 input for Mux10<br>10 : Select .2 input for Mux10<br>11 : Select .3 input for Mux10<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-39. TRIP8MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19-18 | MUX9 | R/W | 0h | Select Bits for EPWM-XBAR TRIP8 Mux9:<br>00 : Select .0 input for Mux9<br>01 : Select .1 input for Mux9<br>10 : Select .2 input for Mux9<br>11 : Select .3 input for Mux9<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX8 | R/W | 0h | Select Bits for EPWM-XBAR TRIP8 Mux8:<br>00 : Select .0 input for Mux8<br>01 : Select .1 input for Mux8<br>10 : Select .2 input for Mux8<br>11 : Select .3 input for Mux8<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX7 | R/W | 0h | Select Bits for EPWM-XBAR TRIP8 Mux7:<br>00 : Select .0 input for Mux7<br>01 : Select .1 input for Mux7<br>10 : Select .2 input for Mux7<br>11 : Select .3 input for Mux7<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX6 | R/W | 0h | Select Bits for EPWM-XBAR TRIP8 Mux6:<br>00 : Select .0 input for Mux6<br>01 : Select .1 input for Mux6<br>10 : Select .2 input for Mux6<br>11 : Select .3 input for Mux6<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX5 | R/W | 0h | Select Bits for EPWM-XBAR TRIP8 Mux5:<br>00 : Select .0 input for Mux5<br>01 : Select .1 input for Mux5<br>10 : Select .2 input for Mux5<br>11 : Select .3 input for Mux5<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX4 | R/W | 0h | Select Bits for EPWM-XBAR TRIP8 Mux4:<br>00 : Select .0 input for Mux4<br>01 : Select .1 input for Mux4<br>10 : Select .2 input for Mux4<br>11 : Select .3 input for Mux4<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7-6 | MUX3 | R/W | 0h | Select Bits for EPWM-XBAR TRIP8 Mux3:<br>00 : Select .0 input for Mux3<br>01 : Select .1 input for Mux3<br>10 : Select .2 input for Mux3<br>11 : Select .3 input for Mux3<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX2 | R/W | 0h | Select Bits for EPWM-XBAR TRIP8 Mux2:<br>00 : Select .0 input for Mux2<br>01 : Select .1 input for Mux2<br>10 : Select .2 input for Mux2<br>11 : Select .3 input for Mux2<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-39. TRIP8MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-2 | MUX1 | R/W | 0h | Select Bits for EPWM-XBAR TRIP8 Mux1:<br>00 : Select .0 input for Mux1<br>01 : Select .1 input for Mux1<br>10 : Select .2 input for Mux1<br>11 : Select .3 input for Mux1<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX0 | R/W | 0h | Select Bits for EPWM-XBAR TRIP8 Mux0:<br>00 : Select .0 input for Mux0<br>01 : Select .1 input for Mux0<br>10 : Select .2 input for Mux0<br>11 : Select .3 input for Mux0<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.4.8 TRIP8MUX16TO31CFG Register (Offset = Eh) [Reset = 00000000h]

TRIP8MUX16TO31CFG is shown in Figure 9-35 and described in Table 9-40.

Return to the Summary Table.

ePWM XBAR Mux Configuration for TRIP8

**Figure 9-35. 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 9-40. TRIP8MUX16TO31CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX31 | R/W | 0h | Select Bits for EPWM-XBAR TRIP8 Mux31:<br>00 : Select .0 input for Mux31<br>01 : Select .1 input for Mux31<br>10 : Select .2 input for Mux31<br>11 : Select .3 input for Mux31<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX30 | R/W | 0h | Select Bits for EPWM-XBAR TRIP8 Mux30:<br>00 : Select .0 input for Mux30<br>01 : Select .1 input for Mux30<br>10 : Select .2 input for Mux30<br>11 : Select .3 input for Mux30<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX29 | R/W | 0h | Select Bits for EPWM-XBAR TRIP8 Mux29:<br>00 : Select .0 input for Mux29<br>01 : Select .1 input for Mux29<br>10 : Select .2 input for Mux29<br>11 : Select .3 input for Mux29<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX28 | R/W | 0h | Select Bits for EPWM-XBAR TRIP8 Mux28:<br>00 : Select .0 input for Mux28<br>01 : Select .1 input for Mux28<br>10 : Select .2 input for Mux28<br>11 : Select .3 input for Mux28<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX27 | R/W | 0h | Select Bits for EPWM-XBAR TRIP8 Mux27:<br>00 : Select .0 input for Mux27<br>01 : Select .1 input for Mux27<br>10 : Select .2 input for Mux27<br>11 : Select .3 input for Mux27<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21-20 | MUX26 | R/W | 0h | Select Bits for EPWM-XBAR TRIP8 Mux26:<br>00 : Select .0 input for Mux26<br>01 : Select .1 input for Mux26<br>10 : Select .2 input for Mux26<br>11 : Select .3 input for Mux26<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-40. TRIP8MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19-18 | MUX25 | R/W | 0h | Select Bits for EPWM-XBAR TRIP8 Mux25:<br>00 : Select .0 input for Mux25<br>01 : Select .1 input for Mux25<br>10 : Select .2 input for Mux25<br>11 : Select .3 input for Mux25<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX24 | R/W | 0h | Select Bits for EPWM-XBAR TRIP8 Mux24:<br>00 : Select .0 input for Mux24<br>01 : Select .1 input for Mux24<br>10 : Select .2 input for Mux24<br>11 : Select .3 input for Mux24<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX23 | R/W | 0h | Select Bits for EPWM-XBAR TRIP8 Mux23:<br>00 : Select .0 input for Mux23<br>01 : Select .1 input for Mux23<br>10 : Select .2 input for Mux23<br>11 : Select .3 input for Mux23<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX22 | R/W | 0h | Select Bits for EPWM-XBAR TRIP8 Mux22:<br>00 : Select .0 input for Mux22<br>01 : Select .1 input for Mux22<br>10 : Select .2 input for Mux22<br>11 : Select .3 input for Mux22<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX21 | R/W | 0h | Select Bits for EPWM-XBAR TRIP8 Mux21:<br>00 : Select .0 input for Mux21<br>01 : Select .1 input for Mux21<br>10 : Select .2 input for Mux21<br>11 : Select .3 input for Mux21<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX20 | R/W | 0h | Select Bits for EPWM-XBAR TRIP8 Mux20:<br>00 : Select .0 input for Mux20<br>01 : Select .1 input for Mux20<br>10 : Select .2 input for Mux20<br>11 : Select .3 input for Mux20<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7-6 | MUX19 | R/W | 0h | Select Bits for EPWM-XBAR TRIP8 Mux19:<br>00 : Select .0 input for Mux19<br>01 : Select .1 input for Mux19<br>10 : Select .2 input for Mux19<br>11 : Select .3 input for Mux19<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX18 | R/W | 0h | Select Bits for EPWM-XBAR TRIP8 Mux18:<br>00 : Select .0 input for Mux18<br>01 : Select .1 input for Mux18<br>10 : Select .2 input for Mux18<br>11 : Select .3 input for Mux18<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-40. TRIP8MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-2 | MUX17 | R/W | 0h | Select Bits for EPWM-XBAR TRIP8 Mux17:<br>00 : Select .0 input for Mux17<br>01 : Select .1 input for Mux17<br>10 : Select .2 input for Mux17<br>11 : Select .3 input for Mux17<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX16 | R/W | 0h | Select Bits for EPWM-XBAR TRIP8 Mux16:<br>00 : Select .0 input for Mux16<br>01 : Select .1 input for Mux16<br>10 : Select .2 input for Mux16<br>11 : Select .3 input for Mux16<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.4.9 TRIP9MUX0TO15CFG Register (Offset = 10h) [Reset = 00000000h]

TRIP9MUX0TO15CFG is shown in Figure 9-36 and described in Table 9-41.

Return to the Summary Table.

ePWM XBAR Mux Configuration for TRIP9

**Figure 9-36. 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 9-41. TRIP9MUX0TO15CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX15 | R/W | 0h | Select Bits for EPWM-XBAR TRIP9 Mux15:<br>00 : Select .0 input for Mux15<br>01 : Select .1 input for Mux15<br>10 : Select .2 input for Mux15<br>11 : Select .3 input for Mux15<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX14 | R/W | 0h | Select Bits for EPWM-XBAR TRIP9 Mux14:<br>00 : Select .0 input for Mux14<br>01 : Select .1 input for Mux14<br>10 : Select .2 input for Mux14<br>11 : Select .3 input for Mux14<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX13 | R/W | 0h | Select Bits for EPWM-XBAR TRIP9 Mux13:<br>00 : Select .0 input for Mux13<br>01 : Select .1 input for Mux13<br>10 : Select .2 input for Mux13<br>11 : Select .3 input for Mux13<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX12 | R/W | 0h | Select Bits for EPWM-XBAR TRIP9 Mux12:<br>00 : Select .0 input for Mux12<br>01 : Select .1 input for Mux12<br>10 : Select .2 input for Mux12<br>11 : Select .3 input for Mux12<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX11 | R/W | 0h | Select Bits for EPWM-XBAR TRIP9 Mux11:<br>00 : Select .0 input for Mux11<br>01 : Select .1 input for Mux11<br>10 : Select .2 input for Mux11<br>11 : Select .3 input for Mux11<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21-20 | MUX10 | R/W | 0h | Select Bits for EPWM-XBAR TRIP9 Mux10:<br>00 : Select .0 input for Mux10<br>01 : Select .1 input for Mux10<br>10 : Select .2 input for Mux10<br>11 : Select .3 input for Mux10<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-41. TRIP9MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19-18 | MUX9 | R/W | 0h | Select Bits for EPWM-XBAR TRIP9 Mux9:<br>00 : Select .0 input for Mux9<br>01 : Select .1 input for Mux9<br>10 : Select .2 input for Mux9<br>11 : Select .3 input for Mux9<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX8 | R/W | 0h | Select Bits for EPWM-XBAR TRIP9 Mux8:<br>00 : Select .0 input for Mux8<br>01 : Select .1 input for Mux8<br>10 : Select .2 input for Mux8<br>11 : Select .3 input for Mux8<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX7 | R/W | 0h | Select Bits for EPWM-XBAR TRIP9 Mux7:<br>00 : Select .0 input for Mux7<br>01 : Select .1 input for Mux7<br>10 : Select .2 input for Mux7<br>11 : Select .3 input for Mux7<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX6 | R/W | 0h | Select Bits for EPWM-XBAR TRIP9 Mux6:<br>00 : Select .0 input for Mux6<br>01 : Select .1 input for Mux6<br>10 : Select .2 input for Mux6<br>11 : Select .3 input for Mux6<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX5 | R/W | 0h | Select Bits for EPWM-XBAR TRIP9 Mux5:<br>00 : Select .0 input for Mux5<br>01 : Select .1 input for Mux5<br>10 : Select .2 input for Mux5<br>11 : Select .3 input for Mux5<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX4 | R/W | 0h | Select Bits for EPWM-XBAR TRIP9 Mux4:<br>00 : Select .0 input for Mux4<br>01 : Select .1 input for Mux4<br>10 : Select .2 input for Mux4<br>11 : Select .3 input for Mux4<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7-6 | MUX3 | R/W | 0h | Select Bits for EPWM-XBAR TRIP9 Mux3:<br>00 : Select .0 input for Mux3<br>01 : Select .1 input for Mux3<br>10 : Select .2 input for Mux3<br>11 : Select .3 input for Mux3<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX2 | R/W | 0h | Select Bits for EPWM-XBAR TRIP9 Mux2:<br>00 : Select .0 input for Mux2<br>01 : Select .1 input for Mux2<br>10 : Select .2 input for Mux2<br>11 : Select .3 input for Mux2<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-41. TRIP9MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-2 | MUX1 | R/W | 0h | Select Bits for EPWM-XBAR TRIP9 Mux1:<br>00 : Select .0 input for Mux1<br>01 : Select .1 input for Mux1<br>10 : Select .2 input for Mux1<br>11 : Select .3 input for Mux1<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX0 | R/W | 0h | Select Bits for EPWM-XBAR TRIP9 Mux0:<br>00 : Select .0 input for Mux0<br>01 : Select .1 input for Mux0<br>10 : Select .2 input for Mux0<br>11 : Select .3 input for Mux0<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.4.10 TRIP9MUX16TO31CFG Register (Offset = 12h) [Reset = 00000000h]

TRIP9MUX16TO31CFG is shown in Figure 9-37 and described in Table 9-42.

Return to the Summary Table.

ePWM XBAR Mux Configuration for TRIP9

**Figure 9-37. 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 9-42. TRIP9MUX16TO31CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX31 | R/W | 0h | Select Bits for EPWM-XBAR TRIP9 Mux31:<br>00 : Select .0 input for Mux31<br>01 : Select .1 input for Mux31<br>10 : Select .2 input for Mux31<br>11 : Select .3 input for Mux31<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX30 | R/W | 0h | Select Bits for EPWM-XBAR TRIP9 Mux30:<br>00 : Select .0 input for Mux30<br>01 : Select .1 input for Mux30<br>10 : Select .2 input for Mux30<br>11 : Select .3 input for Mux30<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX29 | R/W | 0h | Select Bits for EPWM-XBAR TRIP9 Mux29:<br>00 : Select .0 input for Mux29<br>01 : Select .1 input for Mux29<br>10 : Select .2 input for Mux29<br>11 : Select .3 input for Mux29<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX28 | R/W | 0h | Select Bits for EPWM-XBAR TRIP9 Mux28:<br>00 : Select .0 input for Mux28<br>01 : Select .1 input for Mux28<br>10 : Select .2 input for Mux28<br>11 : Select .3 input for Mux28<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX27 | R/W | 0h | Select Bits for EPWM-XBAR TRIP9 Mux27:<br>00 : Select .0 input for Mux27<br>01 : Select .1 input for Mux27<br>10 : Select .2 input for Mux27<br>11 : Select .3 input for Mux27<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21-20 | MUX26 | R/W | 0h | Select Bits for EPWM-XBAR TRIP9 Mux26:<br>00 : Select .0 input for Mux26<br>01 : Select .1 input for Mux26<br>10 : Select .2 input for Mux26<br>11 : Select .3 input for Mux26<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-42. TRIP9MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19-18 | MUX25 | R/W | 0h | Select Bits for EPWM-XBAR TRIP9 Mux25:<br>00 : Select .0 input for Mux25<br>01 : Select .1 input for Mux25<br>10 : Select .2 input for Mux25<br>11 : Select .3 input for Mux25<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX24 | R/W | 0h | Select Bits for EPWM-XBAR TRIP9 Mux24:<br>00 : Select .0 input for Mux24<br>01 : Select .1 input for Mux24<br>10 : Select .2 input for Mux24<br>11 : Select .3 input for Mux24<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX23 | R/W | 0h | Select Bits for EPWM-XBAR TRIP9 Mux23:<br>00 : Select .0 input for Mux23<br>01 : Select .1 input for Mux23<br>10 : Select .2 input for Mux23<br>11 : Select .3 input for Mux23<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX22 | R/W | 0h | Select Bits for EPWM-XBAR TRIP9 Mux22:<br>00 : Select .0 input for Mux22<br>01 : Select .1 input for Mux22<br>10 : Select .2 input for Mux22<br>11 : Select .3 input for Mux22<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX21 | R/W | 0h | Select Bits for EPWM-XBAR TRIP9 Mux21:<br>00 : Select .0 input for Mux21<br>01 : Select .1 input for Mux21<br>10 : Select .2 input for Mux21<br>11 : Select .3 input for Mux21<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX20 | R/W | 0h | Select Bits for EPWM-XBAR TRIP9 Mux20:<br>00 : Select .0 input for Mux20<br>01 : Select .1 input for Mux20<br>10 : Select .2 input for Mux20<br>11 : Select .3 input for Mux20<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7-6 | MUX19 | R/W | 0h | Select Bits for EPWM-XBAR TRIP9 Mux19:<br>00 : Select .0 input for Mux19<br>01 : Select .1 input for Mux19<br>10 : Select .2 input for Mux19<br>11 : Select .3 input for Mux19<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX18 | R/W | 0h | Select Bits for EPWM-XBAR TRIP9 Mux18:<br>00 : Select .0 input for Mux18<br>01 : Select .1 input for Mux18<br>10 : Select .2 input for Mux18<br>11 : Select .3 input for Mux18<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-42. TRIP9MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-2 | MUX17 | R/W | 0h | Select Bits for EPWM-XBAR TRIP9 Mux17:<br>00 : Select .0 input for Mux17<br>01 : Select .1 input for Mux17<br>10 : Select .2 input for Mux17<br>11 : Select .3 input for Mux17<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX16 | R/W | 0h | Select Bits for EPWM-XBAR TRIP9 Mux16:<br>00 : Select .0 input for Mux16<br>01 : Select .1 input for Mux16<br>10 : Select .2 input for Mux16<br>11 : Select .3 input for Mux16<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.4.11 TRIP10MUX0TO15CFG Register (Offset = 14h) [Reset = 00000000h]

TRIP10MUX0TO15CFG is shown in Figure 9-38 and described in Table 9-43.

Return to the Summary Table.

ePWM XBAR Mux Configuration for TRIP10

**Figure 9-38. 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 9-43. TRIP10MUX0TO15CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX15 | R/W | 0h | Select Bits for EPWM-XBAR TRIP10 Mux15:<br>00 : Select .0 input for Mux15<br>01 : Select .1 input for Mux15<br>10 : Select .2 input for Mux15<br>11 : Select .3 input for Mux15<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX14 | R/W | 0h | Select Bits for EPWM-XBAR TRIP10 Mux14:<br>00 : Select .0 input for Mux14<br>01 : Select .1 input for Mux14<br>10 : Select .2 input for Mux14<br>11 : Select .3 input for Mux14<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX13 | R/W | 0h | Select Bits for EPWM-XBAR TRIP10 Mux13:<br>00 : Select .0 input for Mux13<br>01 : Select .1 input for Mux13<br>10 : Select .2 input for Mux13<br>11 : Select .3 input for Mux13<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX12 | R/W | 0h | Select Bits for EPWM-XBAR TRIP10 Mux12:<br>00 : Select .0 input for Mux12<br>01 : Select .1 input for Mux12<br>10 : Select .2 input for Mux12<br>11 : Select .3 input for Mux12<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX11 | R/W | 0h | Select Bits for EPWM-XBAR TRIP10 Mux11:<br>00 : Select .0 input for Mux11<br>01 : Select .1 input for Mux11<br>10 : Select .2 input for Mux11<br>11 : Select .3 input for Mux11<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21-20 | MUX10 | R/W | 0h | Select Bits for EPWM-XBAR TRIP10 Mux10:<br>00 : Select .0 input for Mux10<br>01 : Select .1 input for Mux10<br>10 : Select .2 input for Mux10<br>11 : Select .3 input for Mux10<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-43. TRIP10MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19-18 | MUX9 | R/W | 0h | Select Bits for EPWM-XBAR TRIP10 Mux9:<br>00 : Select .0 input for Mux9<br>01 : Select .1 input for Mux9<br>10 : Select .2 input for Mux9<br>11 : Select .3 input for Mux9<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX8 | R/W | 0h | Select Bits for EPWM-XBAR TRIP10 Mux8:<br>00 : Select .0 input for Mux8<br>01 : Select .1 input for Mux8<br>10 : Select .2 input for Mux8<br>11 : Select .3 input for Mux8<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX7 | R/W | 0h | Select Bits for EPWM-XBAR TRIP10 Mux7:<br>00 : Select .0 input for Mux7<br>01 : Select .1 input for Mux7<br>10 : Select .2 input for Mux7<br>11 : Select .3 input for Mux7<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX6 | R/W | 0h | Select Bits for EPWM-XBAR TRIP10 Mux6:<br>00 : Select .0 input for Mux6<br>01 : Select .1 input for Mux6<br>10 : Select .2 input for Mux6<br>11 : Select .3 input for Mux6<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX5 | R/W | 0h | Select Bits for EPWM-XBAR TRIP10 Mux5:<br>00 : Select .0 input for Mux5<br>01 : Select .1 input for Mux5<br>10 : Select .2 input for Mux5<br>11 : Select .3 input for Mux5<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX4 | R/W | 0h | Select Bits for EPWM-XBAR TRIP10 Mux4:<br>00 : Select .0 input for Mux4<br>01 : Select .1 input for Mux4<br>10 : Select .2 input for Mux4<br>11 : Select .3 input for Mux4<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7-6 | MUX3 | R/W | 0h | Select Bits for EPWM-XBAR TRIP10 Mux3:<br>00 : Select .0 input for Mux3<br>01 : Select .1 input for Mux3<br>10 : Select .2 input for Mux3<br>11 : Select .3 input for Mux3<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX2 | R/W | 0h | Select Bits for EPWM-XBAR TRIP10 Mux2:<br>00 : Select .0 input for Mux2<br>01 : Select .1 input for Mux2<br>10 : Select .2 input for Mux2<br>11 : Select .3 input for Mux2<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-43. TRIP10MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-2 | MUX1 | R/W | 0h | Select Bits for EPWM-XBAR TRIP10 Mux1:<br>00 : Select .0 input for Mux1<br>01 : Select .1 input for Mux1<br>10 : Select .2 input for Mux1<br>11 : Select .3 input for Mux1<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX0 | R/W | 0h | Select Bits for EPWM-XBAR TRIP10 Mux0:<br>00 : Select .0 input for Mux0<br>01 : Select .1 input for Mux0<br>10 : Select .2 input for Mux0<br>11 : Select .3 input for Mux0<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.4.12 TRIP10MUX16TO31CFG Register (Offset = 16h) [Reset = 00000000h]

TRIP10MUX16TO31CFG is shown in Figure 9-39 and described in Table 9-44.

Return to the Summary Table.

ePWM XBAR Mux Configuration for TRIP10

**Figure 9-39. 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 9-44. TRIP10MUX16TO31CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX31 | R/W | 0h | Select Bits for EPWM-XBAR TRIP10 Mux31:<br>00 : Select .0 input for Mux31<br>01 : Select .1 input for Mux31<br>10 : Select .2 input for Mux31<br>11 : Select .3 input for Mux31<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX30 | R/W | 0h | Select Bits for EPWM-XBAR TRIP10 Mux30:<br>00 : Select .0 input for Mux30<br>01 : Select .1 input for Mux30<br>10 : Select .2 input for Mux30<br>11 : Select .3 input for Mux30<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX29 | R/W | 0h | Select Bits for EPWM-XBAR TRIP10 Mux29:<br>00 : Select .0 input for Mux29<br>01 : Select .1 input for Mux29<br>10 : Select .2 input for Mux29<br>11 : Select .3 input for Mux29<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX28 | R/W | 0h | Select Bits for EPWM-XBAR TRIP10 Mux28:<br>00 : Select .0 input for Mux28<br>01 : Select .1 input for Mux28<br>10 : Select .2 input for Mux28<br>11 : Select .3 input for Mux28<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX27 | R/W | 0h | Select Bits for EPWM-XBAR TRIP10 Mux27:<br>00 : Select .0 input for Mux27<br>01 : Select .1 input for Mux27<br>10 : Select .2 input for Mux27<br>11 : Select .3 input for Mux27<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21-20 | MUX26 | R/W | 0h | Select Bits for EPWM-XBAR TRIP10 Mux26:<br>00 : Select .0 input for Mux26<br>01 : Select .1 input for Mux26<br>10 : Select .2 input for Mux26<br>11 : Select .3 input for Mux26<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-44. TRIP10MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19-18 | MUX25 | R/W | 0h | Select Bits for EPWM-XBAR TRIP10 Mux25:<br>00 : Select .0 input for Mux25<br>01 : Select .1 input for Mux25<br>10 : Select .2 input for Mux25<br>11 : Select .3 input for Mux25<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX24 | R/W | 0h | Select Bits for EPWM-XBAR TRIP10 Mux24:<br>00 : Select .0 input for Mux24<br>01 : Select .1 input for Mux24<br>10 : Select .2 input for Mux24<br>11 : Select .3 input for Mux24<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX23 | R/W | 0h | Select Bits for EPWM-XBAR TRIP10 Mux23:<br>00 : Select .0 input for Mux23<br>01 : Select .1 input for Mux23<br>10 : Select .2 input for Mux23<br>11 : Select .3 input for Mux23<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX22 | R/W | 0h | Select Bits for EPWM-XBAR TRIP10 Mux22:<br>00 : Select .0 input for Mux22<br>01 : Select .1 input for Mux22<br>10 : Select .2 input for Mux22<br>11 : Select .3 input for Mux22<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX21 | R/W | 0h | Select Bits for EPWM-XBAR TRIP10 Mux21:<br>00 : Select .0 input for Mux21<br>01 : Select .1 input for Mux21<br>10 : Select .2 input for Mux21<br>11 : Select .3 input for Mux21<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX20 | R/W | 0h | Select Bits for EPWM-XBAR TRIP10 Mux20:<br>00 : Select .0 input for Mux20<br>01 : Select .1 input for Mux20<br>10 : Select .2 input for Mux20<br>11 : Select .3 input for Mux20<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7-6 | MUX19 | R/W | 0h | Select Bits for EPWM-XBAR TRIP10 Mux19:<br>00 : Select .0 input for Mux19<br>01 : Select .1 input for Mux19<br>10 : Select .2 input for Mux19<br>11 : Select .3 input for Mux19<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX18 | R/W | 0h | Select Bits for EPWM-XBAR TRIP10 Mux18:<br>00 : Select .0 input for Mux18<br>01 : Select .1 input for Mux18<br>10 : Select .2 input for Mux18<br>11 : Select .3 input for Mux18<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-44. TRIP10MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-2 | MUX17 | R/W | 0h | Select Bits for EPWM-XBAR TRIP10 Mux17:<br>00 : Select .0 input for Mux17<br>01 : Select .1 input for Mux17<br>10 : Select .2 input for Mux17<br>11 : Select .3 input for Mux17<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX16 | R/W | 0h | Select Bits for EPWM-XBAR TRIP10 Mux16:<br>00 : Select .0 input for Mux16<br>01 : Select .1 input for Mux16<br>10 : Select .2 input for Mux16<br>11 : Select .3 input for Mux16<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.4.13 TRIP11MUX0TO15CFG Register (Offset = 18h) [Reset = 00000000h]

TRIP11MUX0TO15CFG is shown in Figure 9-40 and described in Table 9-45.

Return to the Summary Table.

ePWM XBAR Mux Configuration for TRIP11

**Figure 9-40. 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 9-45. TRIP11MUX0TO15CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX15 | R/W | 0h | Select Bits for EPWM-XBAR TRIP11 Mux15:<br>00 : Select .0 input for Mux15<br>01 : Select .1 input for Mux15<br>10 : Select .2 input for Mux15<br>11 : Select .3 input for Mux15<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX14 | R/W | 0h | Select Bits for EPWM-XBAR TRIP11 Mux14:<br>00 : Select .0 input for Mux14<br>01 : Select .1 input for Mux14<br>10 : Select .2 input for Mux14<br>11 : Select .3 input for Mux14<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX13 | R/W | 0h | Select Bits for EPWM-XBAR TRIP11 Mux13:<br>00 : Select .0 input for Mux13<br>01 : Select .1 input for Mux13<br>10 : Select .2 input for Mux13<br>11 : Select .3 input for Mux13<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX12 | R/W | 0h | Select Bits for EPWM-XBAR TRIP11 Mux12:<br>00 : Select .0 input for Mux12<br>01 : Select .1 input for Mux12<br>10 : Select .2 input for Mux12<br>11 : Select .3 input for Mux12<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX11 | R/W | 0h | Select Bits for EPWM-XBAR TRIP11 Mux11:<br>00 : Select .0 input for Mux11<br>01 : Select .1 input for Mux11<br>10 : Select .2 input for Mux11<br>11 : Select .3 input for Mux11<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21-20 | MUX10 | R/W | 0h | Select Bits for EPWM-XBAR TRIP11 Mux10:<br>00 : Select .0 input for Mux10<br>01 : Select .1 input for Mux10<br>10 : Select .2 input for Mux10<br>11 : Select .3 input for Mux10<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-45. TRIP11MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19-18 | MUX9 | R/W | 0h | Select Bits for EPWM-XBAR TRIP11 Mux9:<br>00 : Select .0 input for Mux9<br>01 : Select .1 input for Mux9<br>10 : Select .2 input for Mux9<br>11 : Select .3 input for Mux9<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX8 | R/W | 0h | Select Bits for EPWM-XBAR TRIP11 Mux8:<br>00 : Select .0 input for Mux8<br>01 : Select .1 input for Mux8<br>10 : Select .2 input for Mux8<br>11 : Select .3 input for Mux8<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX7 | R/W | 0h | Select Bits for EPWM-XBAR TRIP11 Mux7:<br>00 : Select .0 input for Mux7<br>01 : Select .1 input for Mux7<br>10 : Select .2 input for Mux7<br>11 : Select .3 input for Mux7<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX6 | R/W | 0h | Select Bits for EPWM-XBAR TRIP11 Mux6:<br>00 : Select .0 input for Mux6<br>01 : Select .1 input for Mux6<br>10 : Select .2 input for Mux6<br>11 : Select .3 input for Mux6<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX5 | R/W | 0h | Select Bits for EPWM-XBAR TRIP11 Mux5:<br>00 : Select .0 input for Mux5<br>01 : Select .1 input for Mux5<br>10 : Select .2 input for Mux5<br>11 : Select .3 input for Mux5<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX4 | R/W | 0h | Select Bits for EPWM-XBAR TRIP11 Mux4:<br>00 : Select .0 input for Mux4<br>01 : Select .1 input for Mux4<br>10 : Select .2 input for Mux4<br>11 : Select .3 input for Mux4<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7-6 | MUX3 | R/W | 0h | Select Bits for EPWM-XBAR TRIP11 Mux3:<br>00 : Select .0 input for Mux3<br>01 : Select .1 input for Mux3<br>10 : Select .2 input for Mux3<br>11 : Select .3 input for Mux3<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX2 | R/W | 0h | Select Bits for EPWM-XBAR TRIP11 Mux2:<br>00 : Select .0 input for Mux2<br>01 : Select .1 input for Mux2<br>10 : Select .2 input for Mux2<br>11 : Select .3 input for Mux2<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-45. TRIP11MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-2 | MUX1 | R/W | 0h | Select Bits for EPWM-XBAR TRIP11 Mux1:<br>00 : Select .0 input for Mux1<br>01 : Select .1 input for Mux1<br>10 : Select .2 input for Mux1<br>11 : Select .3 input for Mux1<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX0 | R/W | 0h | Select Bits for EPWM-XBAR TRIP11 Mux0:<br>00 : Select .0 input for Mux0<br>01 : Select .1 input for Mux0<br>10 : Select .2 input for Mux0<br>11 : Select .3 input for Mux0<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.4.14 TRIP11MUX16TO31CFG Register (Offset = 1Ah) [Reset = 00000000h]

TRIP11MUX16TO31CFG is shown in Figure 9-41 and described in Table 9-46.

Return to the Summary Table.

ePWM XBAR Mux Configuration for TRIP11

**Figure 9-41. 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 9-46. TRIP11MUX16TO31CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX31 | R/W | 0h | Select Bits for EPWM-XBAR TRIP11 Mux31:<br>00 : Select .0 input for Mux31<br>01 : Select .1 input for Mux31<br>10 : Select .2 input for Mux31<br>11 : Select .3 input for Mux31<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX30 | R/W | 0h | Select Bits for EPWM-XBAR TRIP11 Mux30:<br>00 : Select .0 input for Mux30<br>01 : Select .1 input for Mux30<br>10 : Select .2 input for Mux30<br>11 : Select .3 input for Mux30<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX29 | R/W | 0h | Select Bits for EPWM-XBAR TRIP11 Mux29:<br>00 : Select .0 input for Mux29<br>01 : Select .1 input for Mux29<br>10 : Select .2 input for Mux29<br>11 : Select .3 input for Mux29<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX28 | R/W | 0h | Select Bits for EPWM-XBAR TRIP11 Mux28:<br>00 : Select .0 input for Mux28<br>01 : Select .1 input for Mux28<br>10 : Select .2 input for Mux28<br>11 : Select .3 input for Mux28<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX27 | R/W | 0h | Select Bits for EPWM-XBAR TRIP11 Mux27:<br>00 : Select .0 input for Mux27<br>01 : Select .1 input for Mux27<br>10 : Select .2 input for Mux27<br>11 : Select .3 input for Mux27<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21-20 | MUX26 | R/W | 0h | Select Bits for EPWM-XBAR TRIP11 Mux26:<br>00 : Select .0 input for Mux26<br>01 : Select .1 input for Mux26<br>10 : Select .2 input for Mux26<br>11 : Select .3 input for Mux26<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-46. TRIP11MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19-18 | MUX25 | R/W | 0h | Select Bits for EPWM-XBAR TRIP11 Mux25:<br>00 : Select .0 input for Mux25<br>01 : Select .1 input for Mux25<br>10 : Select .2 input for Mux25<br>11 : Select .3 input for Mux25<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX24 | R/W | 0h | Select Bits for EPWM-XBAR TRIP11 Mux24:<br>00 : Select .0 input for Mux24<br>01 : Select .1 input for Mux24<br>10 : Select .2 input for Mux24<br>11 : Select .3 input for Mux24<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX23 | R/W | 0h | Select Bits for EPWM-XBAR TRIP11 Mux23:<br>00 : Select .0 input for Mux23<br>01 : Select .1 input for Mux23<br>10 : Select .2 input for Mux23<br>11 : Select .3 input for Mux23<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX22 | R/W | 0h | Select Bits for EPWM-XBAR TRIP11 Mux22:<br>00 : Select .0 input for Mux22<br>01 : Select .1 input for Mux22<br>10 : Select .2 input for Mux22<br>11 : Select .3 input for Mux22<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX21 | R/W | 0h | Select Bits for EPWM-XBAR TRIP11 Mux21:<br>00 : Select .0 input for Mux21<br>01 : Select .1 input for Mux21<br>10 : Select .2 input for Mux21<br>11 : Select .3 input for Mux21<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX20 | R/W | 0h | Select Bits for EPWM-XBAR TRIP11 Mux20:<br>00 : Select .0 input for Mux20<br>01 : Select .1 input for Mux20<br>10 : Select .2 input for Mux20<br>11 : Select .3 input for Mux20<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7-6 | MUX19 | R/W | 0h | Select Bits for EPWM-XBAR TRIP11 Mux19:<br>00 : Select .0 input for Mux19<br>01 : Select .1 input for Mux19<br>10 : Select .2 input for Mux19<br>11 : Select .3 input for Mux19<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX18 | R/W | 0h | Select Bits for EPWM-XBAR TRIP11 Mux18:<br>00 : Select .0 input for Mux18<br>01 : Select .1 input for Mux18<br>10 : Select .2 input for Mux18<br>11 : Select .3 input for Mux18<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-46. TRIP11MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-2 | MUX17 | R/W | 0h | Select Bits for EPWM-XBAR TRIP11 Mux17:<br>00 : Select .0 input for Mux17<br>01 : Select .1 input for Mux17<br>10 : Select .2 input for Mux17<br>11 : Select .3 input for Mux17<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX16 | R/W | 0h | Select Bits for EPWM-XBAR TRIP11 Mux16:<br>00 : Select .0 input for Mux16<br>01 : Select .1 input for Mux16<br>10 : Select .2 input for Mux16<br>11 : Select .3 input for Mux16<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.4.15 TRIP12MUX0TO15CFG Register (Offset = 1Ch) [Reset = 00000000h]

TRIP12MUX0TO15CFG is shown in Figure 9-42 and described in Table 9-47.

Return to the Summary Table.

ePWM XBAR Mux Configuration for TRIP12

**Figure 9-42. 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 9-47. TRIP12MUX0TO15CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX15 | R/W | 0h | Select Bits for EPWM-XBAR TRIP12 Mux15:<br>00 : Select .0 input for Mux15<br>01 : Select .1 input for Mux15<br>10 : Select .2 input for Mux15<br>11 : Select .3 input for Mux15<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX14 | R/W | 0h | Select Bits for EPWM-XBAR TRIP12 Mux14:<br>00 : Select .0 input for Mux14<br>01 : Select .1 input for Mux14<br>10 : Select .2 input for Mux14<br>11 : Select .3 input for Mux14<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX13 | R/W | 0h | Select Bits for EPWM-XBAR TRIP12 Mux13:<br>00 : Select .0 input for Mux13<br>01 : Select .1 input for Mux13<br>10 : Select .2 input for Mux13<br>11 : Select .3 input for Mux13<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX12 | R/W | 0h | Select Bits for EPWM-XBAR TRIP12 Mux12:<br>00 : Select .0 input for Mux12<br>01 : Select .1 input for Mux12<br>10 : Select .2 input for Mux12<br>11 : Select .3 input for Mux12<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX11 | R/W | 0h | Select Bits for EPWM-XBAR TRIP12 Mux11:<br>00 : Select .0 input for Mux11<br>01 : Select .1 input for Mux11<br>10 : Select .2 input for Mux11<br>11 : Select .3 input for Mux11<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21-20 | MUX10 | R/W | 0h | Select Bits for EPWM-XBAR TRIP12 Mux10:<br>00 : Select .0 input for Mux10<br>01 : Select .1 input for Mux10<br>10 : Select .2 input for Mux10<br>11 : Select .3 input for Mux10<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-47. TRIP12MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19-18 | MUX9 | R/W | 0h | Select Bits for EPWM-XBAR TRIP12 Mux9:<br>00 : Select .0 input for Mux9<br>01 : Select .1 input for Mux9<br>10 : Select .2 input for Mux9<br>11 : Select .3 input for Mux9<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX8 | R/W | 0h | Select Bits for EPWM-XBAR TRIP12 Mux8:<br>00 : Select .0 input for Mux8<br>01 : Select .1 input for Mux8<br>10 : Select .2 input for Mux8<br>11 : Select .3 input for Mux8<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX7 | R/W | 0h | Select Bits for EPWM-XBAR TRIP12 Mux7:<br>00 : Select .0 input for Mux7<br>01 : Select .1 input for Mux7<br>10 : Select .2 input for Mux7<br>11 : Select .3 input for Mux7<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX6 | R/W | 0h | Select Bits for EPWM-XBAR TRIP12 Mux6:<br>00 : Select .0 input for Mux6<br>01 : Select .1 input for Mux6<br>10 : Select .2 input for Mux6<br>11 : Select .3 input for Mux6<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX5 | R/W | 0h | Select Bits for EPWM-XBAR TRIP12 Mux5:<br>00 : Select .0 input for Mux5<br>01 : Select .1 input for Mux5<br>10 : Select .2 input for Mux5<br>11 : Select .3 input for Mux5<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX4 | R/W | 0h | Select Bits for EPWM-XBAR TRIP12 Mux4:<br>00 : Select .0 input for Mux4<br>01 : Select .1 input for Mux4<br>10 : Select .2 input for Mux4<br>11 : Select .3 input for Mux4<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7-6 | MUX3 | R/W | 0h | Select Bits for EPWM-XBAR TRIP12 Mux3:<br>00 : Select .0 input for Mux3<br>01 : Select .1 input for Mux3<br>10 : Select .2 input for Mux3<br>11 : Select .3 input for Mux3<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX2 | R/W | 0h | Select Bits for EPWM-XBAR TRIP12 Mux2:<br>00 : Select .0 input for Mux2<br>01 : Select .1 input for Mux2<br>10 : Select .2 input for Mux2<br>11 : Select .3 input for Mux2<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-47. TRIP12MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-2 | MUX1 | R/W | 0h | Select Bits for EPWM-XBAR TRIP12 Mux1:<br>00 : Select .0 input for Mux1<br>01 : Select .1 input for Mux1<br>10 : Select .2 input for Mux1<br>11 : Select .3 input for Mux1<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX0 | R/W | 0h | Select Bits for EPWM-XBAR TRIP12 Mux0:<br>00 : Select .0 input for Mux0<br>01 : Select .1 input for Mux0<br>10 : Select .2 input for Mux0<br>11 : Select .3 input for Mux0<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.4.16 TRIP12MUX16TO31CFG Register (Offset = 1Eh) [Reset = 00000000h]

TRIP12MUX16TO31CFG is shown in Figure 9-43 and described in Table 9-48.

Return to the Summary Table.

ePWM XBAR Mux Configuration for TRIP12

**Figure 9-43. 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 9-48. TRIP12MUX16TO31CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX31 | R/W | 0h | Select Bits for EPWM-XBAR TRIP12 Mux31:<br>00 : Select .0 input for Mux31<br>01 : Select .1 input for Mux31<br>10 : Select .2 input for Mux31<br>11 : Select .3 input for Mux31<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX30 | R/W | 0h | Select Bits for EPWM-XBAR TRIP12 Mux30:<br>00 : Select .0 input for Mux30<br>01 : Select .1 input for Mux30<br>10 : Select .2 input for Mux30<br>11 : Select .3 input for Mux30<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX29 | R/W | 0h | Select Bits for EPWM-XBAR TRIP12 Mux29:<br>00 : Select .0 input for Mux29<br>01 : Select .1 input for Mux29<br>10 : Select .2 input for Mux29<br>11 : Select .3 input for Mux29<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX28 | R/W | 0h | Select Bits for EPWM-XBAR TRIP12 Mux28:<br>00 : Select .0 input for Mux28<br>01 : Select .1 input for Mux28<br>10 : Select .2 input for Mux28<br>11 : Select .3 input for Mux28<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX27 | R/W | 0h | Select Bits for EPWM-XBAR TRIP12 Mux27:<br>00 : Select .0 input for Mux27<br>01 : Select .1 input for Mux27<br>10 : Select .2 input for Mux27<br>11 : Select .3 input for Mux27<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21-20 | MUX26 | R/W | 0h | Select Bits for EPWM-XBAR TRIP12 Mux26:<br>00 : Select .0 input for Mux26<br>01 : Select .1 input for Mux26<br>10 : Select .2 input for Mux26<br>11 : Select .3 input for Mux26<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-48. TRIP12MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19-18 | MUX25 | R/W | 0h | Select Bits for EPWM-XBAR TRIP12 Mux25:<br>00 : Select .0 input for Mux25<br>01 : Select .1 input for Mux25<br>10 : Select .2 input for Mux25<br>11 : Select .3 input for Mux25<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX24 | R/W | 0h | Select Bits for EPWM-XBAR TRIP12 Mux24:<br>00 : Select .0 input for Mux24<br>01 : Select .1 input for Mux24<br>10 : Select .2 input for Mux24<br>11 : Select .3 input for Mux24<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX23 | R/W | 0h | Select Bits for EPWM-XBAR TRIP12 Mux23:<br>00 : Select .0 input for Mux23<br>01 : Select .1 input for Mux23<br>10 : Select .2 input for Mux23<br>11 : Select .3 input for Mux23<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX22 | R/W | 0h | Select Bits for EPWM-XBAR TRIP12 Mux22:<br>00 : Select .0 input for Mux22<br>01 : Select .1 input for Mux22<br>10 : Select .2 input for Mux22<br>11 : Select .3 input for Mux22<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX21 | R/W | 0h | Select Bits for EPWM-XBAR TRIP12 Mux21:<br>00 : Select .0 input for Mux21<br>01 : Select .1 input for Mux21<br>10 : Select .2 input for Mux21<br>11 : Select .3 input for Mux21<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX20 | R/W | 0h | Select Bits for EPWM-XBAR TRIP12 Mux20:<br>00 : Select .0 input for Mux20<br>01 : Select .1 input for Mux20<br>10 : Select .2 input for Mux20<br>11 : Select .3 input for Mux20<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7-6 | MUX19 | R/W | 0h | Select Bits for EPWM-XBAR TRIP12 Mux19:<br>00 : Select .0 input for Mux19<br>01 : Select .1 input for Mux19<br>10 : Select .2 input for Mux19<br>11 : Select .3 input for Mux19<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX18 | R/W | 0h | Select Bits for EPWM-XBAR TRIP12 Mux18:<br>00 : Select .0 input for Mux18<br>01 : Select .1 input for Mux18<br>10 : Select .2 input for Mux18<br>11 : Select .3 input for Mux18<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-48. TRIP12MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-2 | MUX17 | R/W | 0h | Select Bits for EPWM-XBAR TRIP12 Mux17:<br>00 : Select .0 input for Mux17<br>01 : Select .1 input for Mux17<br>10 : Select .2 input for Mux17<br>11 : Select .3 input for Mux17<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX16 | R/W | 0h | Select Bits for EPWM-XBAR TRIP12 Mux16:<br>00 : Select .0 input for Mux16<br>01 : Select .1 input for Mux16<br>10 : Select .2 input for Mux16<br>11 : Select .3 input for Mux16<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.4.17 TRIP4MUXENABLE Register (Offset = 20h) [Reset = 00000000h]

TRIP4MUXENABLE is shown in Figure 9-44 and described in Table 9-49.

Return to the Summary Table.

ePWM XBAR Mux Enable for TRIP4

**Figure 9-44. TRIP4MUXENABLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MUX31 | MUX30 | MUX29 | MUX28 | MUX27 | MUX26 | MUX25 | MUX24 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MUX23 | MUX22 | MUX21 | MUX20 | MUX19 | MUX18 | MUX17 | MUX16 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MUX15 | MUX14 | MUX13 | MUX12 | MUX11 | MUX10 | MUX9 | MUX8 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | MUX6 | MUX5 | MUX4 | MUX3 | MUX2 | MUX1 | MUX0 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 9-49. 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<br>0: Respective output of Mux31 is disabled to drive the TRIP4 of EPWM-XBAR<br>1: Respective output of Mux31 is enabled to drive the TRIP4 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 30 | MUX30 | R/W | 0h | Selects the output of Mux30 to drive TRIP4 of EPWM-XBAR<br>0: Respective output of Mux30 is disabled to drive the TRIP4 of EPWM-XBAR<br>1: Respective output of Mux30 is enabled to drive the TRIP4 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29 | MUX29 | R/W | 0h | Selects the output of Mux29 to drive TRIP4 of EPWM-XBAR<br>0: Respective output of Mux29 is disabled to drive the TRIP4 of EPWM-XBAR<br>1: Respective output of Mux29 is enabled to drive the TRIP4 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 28 | MUX28 | R/W | 0h | Selects the output of Mux28 to drive TRIP4 of EPWM-XBAR<br>0: Respective output of Mux28 is disabled to drive the TRIP4 of EPWM-XBAR<br>1: Respective output of Mux28 is enabled to drive the TRIP4 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-49. 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<br>0: Respective output of Mux27 is disabled to drive the TRIP4 of EPWM-XBAR<br>1: Respective output of Mux27 is enabled to drive the TRIP4 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 26 | MUX26 | R/W | 0h | Selects the output of Mux26 to drive TRIP4 of EPWM-XBAR<br>0: Respective output of Mux26 is disabled to drive the TRIP4 of EPWM-XBAR<br>1: Respective output of Mux26 is enabled to drive the TRIP4 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25 | MUX25 | R/W | 0h | Selects the output of Mux25 to drive TRIP4 of EPWM-XBAR<br>0: Respective output of Mux25 is disabled to drive the TRIP4 of EPWM-XBAR<br>1: Respective output of Mux25 is enabled to drive the TRIP4 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 24 | MUX24 | R/W | 0h | Selects the output of Mux24 to drive TRIP4 of EPWM-XBAR<br>0: Respective output of Mux24 is disabled to drive the TRIP4 of EPWM-XBAR<br>1: Respective output of Mux24 is enabled to drive the TRIP4 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23 | MUX23 | R/W | 0h | Selects the output of Mux23 to drive TRIP4 of EPWM-XBAR<br>0: Respective output of Mux23 is disabled to drive the TRIP4 of EPWM-XBAR<br>1: Respective output of Mux23 is enabled to drive the TRIP4 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 22 | MUX22 | R/W | 0h | Selects the output of Mux22 to drive TRIP4 of EPWM-XBAR<br>0: Respective output of Mux22 is disabled to drive the TRIP4 of EPWM-XBAR<br>1: Respective output of Mux22 is enabled to drive the TRIP4 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21 | MUX21 | R/W | 0h | Selects the output of Mux21 to drive TRIP4 of EPWM-XBAR<br>0: Respective output of Mux21 is disabled to drive the TRIP4 of EPWM-XBAR<br>1: Respective output of Mux21 is enabled to drive the TRIP4 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 20 | MUX20 | R/W | 0h | Selects the output of Mux20 to drive TRIP4 of EPWM-XBAR<br>0: Respective output of Mux20 is disabled to drive the TRIP4 of EPWM-XBAR<br>1: Respective output of Mux20 is enabled to drive the TRIP4 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-49. 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<br>0: Respective output of Mux19 is disabled to drive the TRIP4 of EPWM-XBAR<br>1: Respective output of Mux19 is enabled to drive the TRIP4 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 18 | MUX18 | R/W | 0h | Selects the output of Mux18 to drive TRIP4 of EPWM-XBAR<br>0: Respective output of Mux18 is disabled to drive the TRIP4 of EPWM-XBAR<br>1: Respective output of Mux18 is enabled to drive the TRIP4 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17 | MUX17 | R/W | 0h | Selects the output of Mux17 to drive TRIP4 of EPWM-XBAR<br>0: Respective output of Mux17 is disabled to drive the TRIP4 of EPWM-XBAR<br>1: Respective output of Mux17 is enabled to drive the TRIP4 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 16 | MUX16 | R/W | 0h | Selects the output of Mux16 to drive TRIP4 of EPWM-XBAR<br>0: Respective output of Mux16 is disabled to drive the TRIP4 of EPWM-XBAR<br>1: Respective output of Mux16 is enabled to drive the TRIP4 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15 | MUX15 | R/W | 0h | Selects the output of Mux15 to drive TRIP4 of EPWM-XBAR<br>0: Respective output of Mux15 is disabled to drive the TRIP4 of EPWM-XBAR<br>1: Respective output of Mux15 is enabled to drive the TRIP4 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 14 | MUX14 | R/W | 0h | Selects the output of Mux14 to drive TRIP4 of EPWM-XBAR<br>0: Respective output of Mux14 is disabled to drive the TRIP4 of EPWM-XBAR<br>1: Respective output of Mux14 is enabled to drive the TRIP4 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13 | MUX13 | R/W | 0h | Selects the output of Mux13 to drive TRIP4 of EPWM-XBAR<br>0: Respective output of Mux13 is disabled to drive the TRIP4 of EPWM-XBAR<br>1: Respective output of Mux13 is enabled to drive the TRIP4 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 12 | MUX12 | R/W | 0h | Selects the output of Mux12 to drive TRIP4 of EPWM-XBAR<br>0: Respective output of Mux12 is disabled to drive the TRIP4 of EPWM-XBAR<br>1: Respective output of Mux12 is enabled to drive the TRIP4 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-49. 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<br>0: Respective output of Mux11 is disabled to drive the TRIP4 of EPWM-XBAR<br>1: Respective output of Mux11 is enabled to drive the TRIP4 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 10 | MUX10 | R/W | 0h | Selects the output of Mux10 to drive TRIP4 of EPWM-XBAR<br>0: Respective output of Mux10 is disabled to drive the TRIP4 of EPWM-XBAR<br>1: Respective output of Mux10 is enabled to drive the TRIP4 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9 | MUX9 | R/W | 0h | Selects the output of Mux9 to drive TRIP4 of EPWM-XBAR<br>0: Respective output of Mux9 is disabled to drive the TRIP4 of EPWM-XBAR<br>1: Respective output of Mux9 is enabled to drive the TRIP4 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 8 | MUX8 | R/W | 0h | Selects the output of Mux8 to drive TRIP4 of EPWM-XBAR<br>0: Respective output of Mux8 is disabled to drive the TRIP4 of EPWM-XBAR<br>1: Respective output of Mux8 is enabled to drive the TRIP4 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7 | MUX7 | R/W | 0h | Selects the output of Mux7 to drive TRIP4 of EPWM-XBAR<br>0: Respective output of Mux7 is disabled to drive the TRIP4 of EPWM-XBAR<br>1: Respective output of Mux7 is enabled to drive the TRIP4 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 6 | MUX6 | R/W | 0h | Selects the output of Mux6 to drive TRIP4 of EPWM-XBAR<br>0: Respective output of Mux6 is disabled to drive the TRIP4 of EPWM-XBAR<br>1: Respective output of Mux6 is enabled to drive the TRIP4 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5 | MUX5 | R/W | 0h | Selects the output of Mux5 to drive TRIP4 of EPWM-XBAR<br>0: Respective output of Mux5 is disabled to drive the TRIP4 of EPWM-XBAR<br>1: Respective output of Mux5 is enabled to drive the TRIP4 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 4 | MUX4 | R/W | 0h | Selects the output of Mux4 to drive TRIP4 of EPWM-XBAR<br>0: Respective output of Mux4 is disabled to drive the TRIP4 of EPWM-XBAR<br>1: Respective output of Mux4 is enabled to drive the TRIP4 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-49. 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<br>0: Respective output of Mux3 is disabled to drive the TRIP4 of EPWM-XBAR<br>1: Respective output of Mux3 is enabled to drive the TRIP4 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 2 | MUX2 | R/W | 0h | Selects the output of Mux2 to drive TRIP4 of EPWM-XBAR<br>0: Respective output of Mux2 is disabled to drive the TRIP4 of EPWM-XBAR<br>1: Respective output of Mux2 is enabled to drive the TRIP4 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1 | MUX1 | R/W | 0h | Selects the output of Mux1 to drive TRIP4 of EPWM-XBAR<br>0: Respective output of Mux1 is disabled to drive the TRIP4 of EPWM-XBAR<br>1: Respective output of Mux1 is enabled to drive the TRIP4 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 0 | MUX0 | R/W | 0h | Selects the output of Mux0 to drive TRIP4 of EPWM-XBAR<br>0: Respective output of Mux0 is disabled to drive the TRIP4 of EPWM-XBAR<br>1: Respective output of Mux0 is enabled to drive the TRIP4 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.4.18 TRIP5MUXENABLE Register (Offset = 22h) [Reset = 00000000h]

TRIP5MUXENABLE is shown in Figure 9-45 and described in Table 9-50.

Return to the Summary Table.

ePWM XBAR Mux Enable for TRIP5

**Figure 9-45. TRIP5MUXENABLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MUX31 | MUX30 | MUX29 | MUX28 | MUX27 | MUX26 | MUX25 | MUX24 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MUX23 | MUX22 | MUX21 | MUX20 | MUX19 | MUX18 | MUX17 | MUX16 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MUX15 | MUX14 | MUX13 | MUX12 | MUX11 | MUX10 | MUX9 | MUX8 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | MUX6 | MUX5 | MUX4 | MUX3 | MUX2 | MUX1 | MUX0 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 9-50. 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<br>0: Respective output of Mux31 is disabled to drive the TRIP5 of EPWM-XBAR<br>1: Respective output of Mux31 is enabled to drive the TRIP5 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 30 | MUX30 | R/W | 0h | Selects the output of Mux30 to drive TRIP5 of EPWM-XBAR<br>0: Respective output of Mux30 is disabled to drive the TRIP5 of EPWM-XBAR<br>1: Respective output of Mux30 is enabled to drive the TRIP5 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29 | MUX29 | R/W | 0h | Selects the output of Mux29 to drive TRIP5 of EPWM-XBAR<br>0: Respective output of Mux29 is disabled to drive the TRIP5 of EPWM-XBAR<br>1: Respective output of Mux29 is enabled to drive the TRIP5 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 28 | MUX28 | R/W | 0h | Selects the output of Mux28 to drive TRIP5 of EPWM-XBAR<br>0: Respective output of Mux28 is disabled to drive the TRIP5 of EPWM-XBAR<br>1: Respective output of Mux28 is enabled to drive the TRIP5 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-50. 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<br>0: Respective output of Mux27 is disabled to drive the TRIP5 of EPWM-XBAR<br>1: Respective output of Mux27 is enabled to drive the TRIP5 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 26 | MUX26 | R/W | 0h | Selects the output of Mux26 to drive TRIP5 of EPWM-XBAR<br>0: Respective output of Mux26 is disabled to drive the TRIP5 of EPWM-XBAR<br>1: Respective output of Mux26 is enabled to drive the TRIP5 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25 | MUX25 | R/W | 0h | Selects the output of Mux25 to drive TRIP5 of EPWM-XBAR<br>0: Respective output of Mux25 is disabled to drive the TRIP5 of EPWM-XBAR<br>1: Respective output of Mux25 is enabled to drive the TRIP5 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 24 | MUX24 | R/W | 0h | Selects the output of Mux24 to drive TRIP5 of EPWM-XBAR<br>0: Respective output of Mux24 is disabled to drive the TRIP5 of EPWM-XBAR<br>1: Respective output of Mux24 is enabled to drive the TRIP5 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23 | MUX23 | R/W | 0h | Selects the output of Mux23 to drive TRIP5 of EPWM-XBAR<br>0: Respective output of Mux23 is disabled to drive the TRIP5 of EPWM-XBAR<br>1: Respective output of Mux23 is enabled to drive the TRIP5 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 22 | MUX22 | R/W | 0h | Selects the output of Mux22 to drive TRIP5 of EPWM-XBAR<br>0: Respective output of Mux22 is disabled to drive the TRIP5 of EPWM-XBAR<br>1: Respective output of Mux22 is enabled to drive the TRIP5 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21 | MUX21 | R/W | 0h | Selects the output of Mux21 to drive TRIP5 of EPWM-XBAR<br>0: Respective output of Mux21 is disabled to drive the TRIP5 of EPWM-XBAR<br>1: Respective output of Mux21 is enabled to drive the TRIP5 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 20 | MUX20 | R/W | 0h | Selects the output of Mux20 to drive TRIP5 of EPWM-XBAR<br>0: Respective output of Mux20 is disabled to drive the TRIP5 of EPWM-XBAR<br>1: Respective output of Mux20 is enabled to drive the TRIP5 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-50. 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<br>0: Respective output of Mux19 is disabled to drive the TRIP5 of EPWM-XBAR<br>1: Respective output of Mux19 is enabled to drive the TRIP5 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 18 | MUX18 | R/W | 0h | Selects the output of Mux18 to drive TRIP5 of EPWM-XBAR<br>0: Respective output of Mux18 is disabled to drive the TRIP5 of EPWM-XBAR<br>1: Respective output of Mux18 is enabled to drive the TRIP5 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17 | MUX17 | R/W | 0h | Selects the output of Mux17 to drive TRIP5 of EPWM-XBAR<br>0: Respective output of Mux17 is disabled to drive the TRIP5 of EPWM-XBAR<br>1: Respective output of Mux17 is enabled to drive the TRIP5 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 16 | MUX16 | R/W | 0h | Selects the output of Mux16 to drive TRIP5 of EPWM-XBAR<br>0: Respective output of Mux16 is disabled to drive the TRIP5 of EPWM-XBAR<br>1: Respective output of Mux16 is enabled to drive the TRIP5 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15 | MUX15 | R/W | 0h | Selects the output of Mux15 to drive TRIP5 of EPWM-XBAR<br>0: Respective output of Mux15 is disabled to drive the TRIP5 of EPWM-XBAR<br>1: Respective output of Mux15 is enabled to drive the TRIP5 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 14 | MUX14 | R/W | 0h | Selects the output of Mux14 to drive TRIP5 of EPWM-XBAR<br>0: Respective output of Mux14 is disabled to drive the TRIP5 of EPWM-XBAR<br>1: Respective output of Mux14 is enabled to drive the TRIP5 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13 | MUX13 | R/W | 0h | Selects the output of Mux13 to drive TRIP5 of EPWM-XBAR<br>0: Respective output of Mux13 is disabled to drive the TRIP5 of EPWM-XBAR<br>1: Respective output of Mux13 is enabled to drive the TRIP5 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 12 | MUX12 | R/W | 0h | Selects the output of Mux12 to drive TRIP5 of EPWM-XBAR<br>0: Respective output of Mux12 is disabled to drive the TRIP5 of EPWM-XBAR<br>1: Respective output of Mux12 is enabled to drive the TRIP5 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-50. 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<br>0: Respective output of Mux11 is disabled to drive the TRIP5 of EPWM-XBAR<br>1: Respective output of Mux11 is enabled to drive the TRIP5 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 10 | MUX10 | R/W | 0h | Selects the output of Mux10 to drive TRIP5 of EPWM-XBAR<br>0: Respective output of Mux10 is disabled to drive the TRIP5 of EPWM-XBAR<br>1: Respective output of Mux10 is enabled to drive the TRIP5 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9 | MUX9 | R/W | 0h | Selects the output of Mux9 to drive TRIP5 of EPWM-XBAR<br>0: Respective output of Mux9 is disabled to drive the TRIP5 of EPWM-XBAR<br>1: Respective output of Mux9 is enabled to drive the TRIP5 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 8 | MUX8 | R/W | 0h | Selects the output of Mux8 to drive TRIP5 of EPWM-XBAR<br>0: Respective output of Mux8 is disabled to drive the TRIP5 of EPWM-XBAR<br>1: Respective output of Mux8 is enabled to drive the TRIP5 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7 | MUX7 | R/W | 0h | Selects the output of Mux7 to drive TRIP5 of EPWM-XBAR<br>0: Respective output of Mux7 is disabled to drive the TRIP5 of EPWM-XBAR<br>1: Respective output of Mux7 is enabled to drive the TRIP5 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 6 | MUX6 | R/W | 0h | Selects the output of Mux6 to drive TRIP5 of EPWM-XBAR<br>0: Respective output of Mux6 is disabled to drive the TRIP5 of EPWM-XBAR<br>1: Respective output of Mux6 is enabled to drive the TRIP5 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5 | MUX5 | R/W | 0h | Selects the output of Mux5 to drive TRIP5 of EPWM-XBAR<br>0: Respective output of Mux5 is disabled to drive the TRIP5 of EPWM-XBAR<br>1: Respective output of Mux5 is enabled to drive the TRIP5 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 4 | MUX4 | R/W | 0h | Selects the output of Mux4 to drive TRIP5 of EPWM-XBAR<br>0: Respective output of Mux4 is disabled to drive the TRIP5 of EPWM-XBAR<br>1: Respective output of Mux4 is enabled to drive the TRIP5 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-50. 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<br>0: Respective output of Mux3 is disabled to drive the TRIP5 of EPWM-XBAR<br>1: Respective output of Mux3 is enabled to drive the TRIP5 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 2 | MUX2 | R/W | 0h | Selects the output of Mux2 to drive TRIP5 of EPWM-XBAR<br>0: Respective output of Mux2 is disabled to drive the TRIP5 of EPWM-XBAR<br>1: Respective output of Mux2 is enabled to drive the TRIP5 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1 | MUX1 | R/W | 0h | Selects the output of Mux1 to drive TRIP5 of EPWM-XBAR<br>0: Respective output of Mux1 is disabled to drive the TRIP5 of EPWM-XBAR<br>1: Respective output of Mux1 is enabled to drive the TRIP5 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 0 | MUX0 | R/W | 0h | Selects the output of mux0 to drive TRIP5 of EPWM-XBAR<br>0: Respective output of Mux0 is disabled to drive the TRIP5 of EPWM-XBAR<br>1: Respective output of Mux0 is enabled to drive the TRIP5 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.4.19 TRIP7MUXENABLE Register (Offset = 24h) [Reset = 00000000h]

TRIP7MUXENABLE is shown in Figure 9-46 and described in Table 9-51.

Return to the Summary Table.

ePWM XBAR Mux Enable for TRIP7

**Figure 9-46. TRIP7MUXENABLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MUX31 | MUX30 | MUX29 | MUX28 | MUX27 | MUX26 | MUX25 | MUX24 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MUX23 | MUX22 | MUX21 | MUX20 | MUX19 | MUX18 | MUX17 | MUX16 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MUX15 | MUX14 | MUX13 | MUX12 | MUX11 | MUX10 | MUX9 | MUX8 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | MUX6 | MUX5 | MUX4 | MUX3 | MUX2 | MUX1 | MUX0 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 9-51. 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<br>0: Respective output of Mux31 is disabled to drive the TRIP7 of EPWM-XBAR<br>1: Respective output of Mux31 is enabled to drive the TRIP7 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 30 | MUX30 | R/W | 0h | Selects the output of Mux30 to drive TRIP7 of EPWM-XBAR<br>0: Respective output of Mux30 is disabled to drive the TRIP7 of EPWM-XBAR<br>1: Respective output of Mux30 is enabled to drive the TRIP7 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29 | MUX29 | R/W | 0h | Selects the output of Mux29 to drive TRIP7 of EPWM-XBAR<br>0: Respective output of Mux29 is disabled to drive the TRIP7 of EPWM-XBAR<br>1: Respective output of Mux29 is enabled to drive the TRIP7 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 28 | MUX28 | R/W | 0h | Selects the output of Mux28 to drive TRIP7 of EPWM-XBAR<br>0: Respective output of Mux28 is disabled to drive the TRIP7 of EPWM-XBAR<br>1: Respective output of Mux28 is enabled to drive the TRIP7 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-51. 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<br>0: Respective output of Mux27 is disabled to drive the TRIP7 of EPWM-XBAR<br>1: Respective output of Mux27 is enabled to drive the TRIP7 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 26 | MUX26 | R/W | 0h | Selects the output of Mux26 to drive TRIP7 of EPWM-XBAR<br>0: Respective output of Mux26 is disabled to drive the TRIP7 of EPWM-XBAR<br>1: Respective output of Mux26 is enabled to drive the TRIP7 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25 | MUX25 | R/W | 0h | Selects the output of Mux25 to drive TRIP7 of EPWM-XBAR<br>0: Respective output of Mux25 is disabled to drive the TRIP7 of EPWM-XBAR<br>1: Respective output of Mux25 is enabled to drive the TRIP7 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 24 | MUX24 | R/W | 0h | Selects the output of Mux24 to drive TRIP7 of EPWM-XBAR<br>0: Respective output of Mux24 is disabled to drive the TRIP7 of EPWM-XBAR<br>1: Respective output of Mux24 is enabled to drive the TRIP7 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23 | MUX23 | R/W | 0h | Selects the output of Mux23 to drive TRIP7 of EPWM-XBAR<br>0: Respective output of Mux23 is disabled to drive the TRIP7 of EPWM-XBAR<br>1: Respective output of Mux23 is enabled to drive the TRIP7 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 22 | MUX22 | R/W | 0h | Selects the output of Mux22 to drive TRIP7 of EPWM-XBAR<br>0: Respective output of Mux22 is disabled to drive the TRIP7 of EPWM-XBAR<br>1: Respective output of Mux22 is enabled to drive the TRIP7 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21 | MUX21 | R/W | 0h | Selects the output of Mux21 to drive TRIP7 of EPWM-XBAR<br>0: Respective output of Mux21 is disabled to drive the TRIP7 of EPWM-XBAR<br>1: Respective output of Mux21 is enabled to drive the TRIP7 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 20 | MUX20 | R/W | 0h | Selects the output of Mux20 to drive TRIP7 of EPWM-XBAR<br>0: Respective output of Mux20 is disabled to drive the TRIP7 of EPWM-XBAR<br>1: Respective output of Mux20 is enabled to drive the TRIP7 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-51. 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<br>0: Respective output of Mux19 is disabled to drive the TRIP7 of EPWM-XBAR<br>1: Respective output of Mux19 is enabled to drive the TRIP7 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 18 | MUX18 | R/W | 0h | Selects the output of Mux18 to drive TRIP7 of EPWM-XBAR<br>0: Respective output of Mux18 is disabled to drive the TRIP7 of EPWM-XBAR<br>1: Respective output of Mux18 is enabled to drive the TRIP7 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17 | MUX17 | R/W | 0h | Selects the output of Mux17 to drive TRIP7 of EPWM-XBAR<br>0: Respective output of Mux17 is disabled to drive the TRIP7 of EPWM-XBAR<br>1: Respective output of Mux17 is enabled to drive the TRIP7 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 16 | MUX16 | R/W | 0h | Selects the output of Mux16 to drive TRIP7 of EPWM-XBAR<br>0: Respective output of Mux16 is disabled to drive the TRIP7 of EPWM-XBAR<br>1: Respective output of Mux16 is enabled to drive the TRIP7 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15 | MUX15 | R/W | 0h | Selects the output of Mux15 to drive TRIP7 of EPWM-XBAR<br>0: Respective output of Mux15 is disabled to drive the TRIP7 of EPWM-XBAR<br>1: Respective output of Mux15 is enabled to drive the TRIP7 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 14 | MUX14 | R/W | 0h | Selects the output of Mux14 to drive TRIP7 of EPWM-XBAR<br>0: Respective output of Mux14 is disabled to drive the TRIP7 of EPWM-XBAR<br>1: Respective output of Mux14 is enabled to drive the TRIP7 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13 | MUX13 | R/W | 0h | Selects the output of Mux13 to drive TRIP7 of EPWM-XBAR<br>0: Respective output of Mux13 is disabled to drive the TRIP7 of EPWM-XBAR<br>1: Respective output of Mux13 is enabled to drive the TRIP7 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 12 | MUX12 | R/W | 0h | Selects the output of Mux12 to drive TRIP7 of EPWM-XBAR<br>0: Respective output of Mux12 is disabled to drive the TRIP7 of EPWM-XBAR<br>1: Respective output of Mux12 is enabled to drive the TRIP7 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-51. 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<br>0: Respective output of Mux11 is disabled to drive the TRIP7 of EPWM-XBAR<br>1: Respective output of Mux11 is enabled to drive the TRIP7 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 10 | MUX10 | R/W | 0h | Selects the output of Mux10 to drive TRIP7 of EPWM-XBAR<br>0: Respective output of Mux10 is disabled to drive the TRIP7 of EPWM-XBAR<br>1: Respective output of Mux10 is enabled to drive the TRIP7 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9 | MUX9 | R/W | 0h | Selects the output of Mux9 to drive TRIP7 of EPWM-XBAR<br>0: Respective output of Mux9 is disabled to drive the TRIP7 of EPWM-XBAR<br>1: Respective output of Mux9 is enabled to drive the TRIP7 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 8 | MUX8 | R/W | 0h | Selects the output of Mux8 to drive TRIP7 of EPWM-XBAR<br>0: Respective output of Mux8 is disabled to drive the TRIP7 of EPWM-XBAR<br>1: Respective output of Mux8 is enabled to drive the TRIP7 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7 | MUX7 | R/W | 0h | Selects the output of Mux7 to drive TRIP7 of EPWM-XBAR<br>0: Respective output of Mux7 is disabled to drive the TRIP7 of EPWM-XBAR<br>1: Respective output of Mux7 is enabled to drive the TRIP7 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 6 | MUX6 | R/W | 0h | Selects the output of Mux6 to drive TRIP7 of EPWM-XBAR<br>0: Respective output of Mux6 is disabled to drive the TRIP7 of EPWM-XBAR<br>1: Respective output of Mux6 is enabled to drive the TRIP7 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5 | MUX5 | R/W | 0h | Selects the output of Mux5 to drive TRIP7 of EPWM-XBAR<br>0: Respective output of Mux5 is disabled to drive the TRIP7 of EPWM-XBAR<br>1: Respective output of Mux5 is enabled to drive the TRIP7 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 4 | MUX4 | R/W | 0h | Selects the output of Mux4 to drive TRIP7 of EPWM-XBAR<br>0: Respective output of Mux4 is disabled to drive the TRIP7 of EPWM-XBAR<br>1: Respective output of Mux4 is enabled to drive the TRIP7 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-51. 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<br>0: Respective output of Mux3 is disabled to drive the TRIP7 of EPWM-XBAR<br>1: Respective output of Mux3 is enabled to drive the TRIP7 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 2 | MUX2 | R/W | 0h | Selects the output of Mux2 to drive TRIP7 of EPWM-XBAR<br>0: Respective output of Mux2 is disabled to drive the TRIP7 of EPWM-XBAR<br>1: Respective output of Mux2 is enabled to drive the TRIP7 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1 | MUX1 | R/W | 0h | Selects the output of Mux1 to drive TRIP7 of EPWM-XBAR<br>0: Respective output of Mux1 is disabled to drive the TRIP7 of EPWM-XBAR<br>1: Respective output of Mux1 is enabled to drive the TRIP7 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 0 | MUX0 | R/W | 0h | Selects the output of mux0 to drive TRIP7 of EPWM-XBAR<br>0: Respective output of Mux0 is disabled to drive the TRIP7 of EPWM-XBAR<br>1: Respective output of Mux0 is enabled to drive the TRIP7 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.4.20 TRIP8MUXENABLE Register (Offset = 26h) [Reset = 00000000h]

TRIP8MUXENABLE is shown in Figure 9-47 and described in Table 9-52.

Return to the Summary Table.

ePWM XBAR Mux Enable for TRIP8

**Figure 9-47. TRIP8MUXENABLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MUX31 | MUX30 | MUX29 | MUX28 | MUX27 | MUX26 | MUX25 | MUX24 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MUX23 | MUX22 | MUX21 | MUX20 | MUX19 | MUX18 | MUX17 | MUX16 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MUX15 | MUX14 | MUX13 | MUX12 | MUX11 | MUX10 | MUX9 | MUX8 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | MUX6 | MUX5 | MUX4 | MUX3 | MUX2 | MUX1 | MUX0 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 9-52. 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<br>0: Respective output of Mux31 is disabled to drive the TRIP8 of EPWM-XBAR<br>1: Respective output of Mux31 is enabled to drive the TRIP8 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 30 | MUX30 | R/W | 0h | Selects the output of Mux30 to drive TRIP8 of EPWM-XBAR<br>0: Respective output of Mux30 is disabled to drive the TRIP8 of EPWM-XBAR<br>1: Respective output of Mux30 is enabled to drive the TRIP8 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29 | MUX29 | R/W | 0h | Selects the output of Mux29 to drive TRIP8 of EPWM-XBAR<br>0: Respective output of Mux29 is disabled to drive the TRIP8 of EPWM-XBAR<br>1: Respective output of Mux29 is enabled to drive the TRIP8 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 28 | MUX28 | R/W | 0h | Selects the output of Mux28 to drive TRIP8 of EPWM-XBAR<br>0: Respective output of Mux28 is disabled to drive the TRIP8 of EPWM-XBAR<br>1: Respective output of Mux28 is enabled to drive the TRIP8 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-52. 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<br>0: Respective output of Mux27 is disabled to drive the TRIP8 of EPWM-XBAR<br>1: Respective output of Mux27 is enabled to drive the TRIP8 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 26 | MUX26 | R/W | 0h | Selects the output of Mux26 to drive TRIP8 of EPWM-XBAR<br>0: Respective output of Mux26 is disabled to drive the TRIP8 of EPWM-XBAR<br>1: Respective output of Mux26 is enabled to drive the TRIP8 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25 | MUX25 | R/W | 0h | Selects the output of Mux25 to drive TRIP8 of EPWM-XBAR<br>0: Respective output of Mux25 is disabled to drive the TRIP8 of EPWM-XBAR<br>1: Respective output of Mux25 is enabled to drive the TRIP8 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 24 | MUX24 | R/W | 0h | Selects the output of Mux24 to drive TRIP8 of EPWM-XBAR<br>0: Respective output of Mux24 is disabled to drive the TRIP8 of EPWM-XBAR<br>1: Respective output of Mux24 is enabled to drive the TRIP8 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23 | MUX23 | R/W | 0h | Selects the output of Mux23 to drive TRIP8 of EPWM-XBAR<br>0: Respective output of Mux23 is disabled to drive the TRIP8 of EPWM-XBAR<br>1: Respective output of Mux23 is enabled to drive the TRIP8 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 22 | MUX22 | R/W | 0h | Selects the output of Mux22 to drive TRIP8 of EPWM-XBAR<br>0: Respective output of Mux22 is disabled to drive the TRIP8 of EPWM-XBAR<br>1: Respective output of Mux22 is enabled to drive the TRIP8 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21 | MUX21 | R/W | 0h | Selects the output of Mux21 to drive TRIP8 of EPWM-XBAR<br>0: Respective output of Mux21 is disabled to drive the TRIP8 of EPWM-XBAR<br>1: Respective output of Mux21 is enabled to drive the TRIP8 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 20 | MUX20 | R/W | 0h | Selects the output of Mux20 to drive TRIP8 of EPWM-XBAR<br>0: Respective output of Mux20 is disabled to drive the TRIP8 of EPWM-XBAR<br>1: Respective output of Mux20 is enabled to drive the TRIP8 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-52. 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<br>0: Respective output of Mux19 is disabled to drive the TRIP8 of EPWM-XBAR<br>1: Respective output of Mux19 is enabled to drive the TRIP8 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 18 | MUX18 | R/W | 0h | Selects the output of Mux18 to drive TRIP8 of EPWM-XBAR<br>0: Respective output of Mux18 is disabled to drive the TRIP8 of EPWM-XBAR<br>1: Respective output of Mux18 is enabled to drive the TRIP8 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17 | MUX17 | R/W | 0h | Selects the output of Mux17 to drive TRIP8 of EPWM-XBAR<br>0: Respective output of Mux17 is disabled to drive the TRIP8 of EPWM-XBAR<br>1: Respective output of Mux17 is enabled to drive the TRIP8 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 16 | MUX16 | R/W | 0h | Selects the output of Mux16 to drive TRIP8 of EPWM-XBAR<br>0: Respective output of Mux16 is disabled to drive the TRIP8 of EPWM-XBAR<br>1: Respective output of Mux16 is enabled to drive the TRIP8 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15 | MUX15 | R/W | 0h | Selects the output of Mux15 to drive TRIP8 of EPWM-XBAR<br>0: Respective output of Mux15 is disabled to drive the TRIP8 of EPWM-XBAR<br>1: Respective output of Mux15 is enabled to drive the TRIP8 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 14 | MUX14 | R/W | 0h | Selects the output of Mux14 to drive TRIP8 of EPWM-XBAR<br>0: Respective output of Mux14 is disabled to drive the TRIP8 of EPWM-XBAR<br>1: Respective output of Mux14 is enabled to drive the TRIP8 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13 | MUX13 | R/W | 0h | Selects the output of Mux13 to drive TRIP8 of EPWM-XBAR<br>0: Respective output of Mux13 is disabled to drive the TRIP8 of EPWM-XBAR<br>1: Respective output of Mux13 is enabled to drive the TRIP8 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 12 | MUX12 | R/W | 0h | Selects the output of Mux12 to drive TRIP8 of EPWM-XBAR<br>0: Respective output of Mux12 is disabled to drive the TRIP8 of EPWM-XBAR<br>1: Respective output of Mux12 is enabled to drive the TRIP8 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-52. 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<br>0: Respective output of Mux11 is disabled to drive the TRIP8 of EPWM-XBAR<br>1: Respective output of Mux11 is enabled to drive the TRIP8 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 10 | MUX10 | R/W | 0h | Selects the output of Mux10 to drive TRIP8 of EPWM-XBAR<br>0: Respective output of Mux10 is disabled to drive the TRIP8 of EPWM-XBAR<br>1: Respective output of Mux10 is enabled to drive the TRIP8 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9 | MUX9 | R/W | 0h | Selects the output of Mux9 to drive TRIP8 of EPWM-XBAR<br>0: Respective output of Mux9 is disabled to drive the TRIP8 of EPWM-XBAR<br>1: Respective output of Mux9 is enabled to drive the TRIP8 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 8 | MUX8 | R/W | 0h | Selects the output of Mux8 to drive TRIP8 of EPWM-XBAR<br>0: Respective output of Mux8 is disabled to drive the TRIP8 of EPWM-XBAR<br>1: Respective output of Mux8 is enabled to drive the TRIP8 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7 | MUX7 | R/W | 0h | Selects the output of Mux7 to drive TRIP8 of EPWM-XBAR<br>0: Respective output of Mux7 is disabled to drive the TRIP8 of EPWM-XBAR<br>1: Respective output of Mux7 is enabled to drive the TRIP8 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 6 | MUX6 | R/W | 0h | Selects the output of Mux6 to drive TRIP8 of EPWM-XBAR<br>0: Respective output of Mux6 is disabled to drive the TRIP8 of EPWM-XBAR<br>1: Respective output of Mux6 is enabled to drive the TRIP8 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5 | MUX5 | R/W | 0h | Selects the output of Mux5 to drive TRIP8 of EPWM-XBAR<br>0: Respective output of Mux5 is disabled to drive the TRIP8 of EPWM-XBAR<br>1: Respective output of Mux5 is enabled to drive the TRIP8 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 4 | MUX4 | R/W | 0h | Selects the output of Mux4 to drive TRIP8 of EPWM-XBAR<br>0: Respective output of Mux4 is disabled to drive the TRIP8 of EPWM-XBAR<br>1: Respective output of Mux4 is enabled to drive the TRIP8 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-52. 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<br>0: Respective output of Mux3 is disabled to drive the TRIP8 of EPWM-XBAR<br>1: Respective output of Mux3 is enabled to drive the TRIP8 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 2 | MUX2 | R/W | 0h | Selects the output of Mux2 to drive TRIP8 of EPWM-XBAR<br>0: Respective output of Mux2 is disabled to drive the TRIP8 of EPWM-XBAR<br>1: Respective output of Mux2 is enabled to drive the TRIP8 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1 | MUX1 | R/W | 0h | Selects the output of Mux1 to drive TRIP8 of EPWM-XBAR<br>0: Respective output of Mux1 is disabled to drive the TRIP8 of EPWM-XBAR<br>1: Respective output of Mux1 is enabled to drive the TRIP8 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 0 | MUX0 | R/W | 0h | Selects the output of mux0 to drive TRIP8 of EPWM-XBAR<br>0: Respective output of Mux0 is disabled to drive the TRIP8 of EPWM-XBAR<br>1: Respective output of Mux0 is enabled to drive the TRIP8 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.4.21 TRIP9MUXENABLE Register (Offset = 28h) [Reset = 00000000h]

TRIP9MUXENABLE is shown in Figure 9-48 and described in Table 9-53.

Return to the Summary Table.

ePWM XBAR Mux Enable for TRIP9

**Figure 9-48. TRIP9MUXENABLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MUX31 | MUX30 | MUX29 | MUX28 | MUX27 | MUX26 | MUX25 | MUX24 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MUX23 | MUX22 | MUX21 | MUX20 | MUX19 | MUX18 | MUX17 | MUX16 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MUX15 | MUX14 | MUX13 | MUX12 | MUX11 | MUX10 | MUX9 | MUX8 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | MUX6 | MUX5 | MUX4 | MUX3 | MUX2 | MUX1 | MUX0 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 9-53. 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<br>0: Respective output of Mux31 is disabled to drive the TRIP9 of EPWM-XBAR<br>1: Respective output of Mux31 is enabled to drive the TRIP9 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 30 | MUX30 | R/W | 0h | Selects the output of Mux30 to drive TRIP9 of EPWM-XBAR<br>0: Respective output of Mux30 is disabled to drive the TRIP9 of EPWM-XBAR<br>1: Respective output of Mux30 is enabled to drive the TRIP9 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29 | MUX29 | R/W | 0h | Selects the output of Mux29 to drive TRIP9 of EPWM-XBAR<br>0: Respective output of Mux29 is disabled to drive the TRIP9 of EPWM-XBAR<br>1: Respective output of Mux29 is enabled to drive the TRIP9 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 28 | MUX28 | R/W | 0h | Selects the output of Mux28 to drive TRIP9 of EPWM-XBAR<br>0: Respective output of Mux28 is disabled to drive the TRIP9 of EPWM-XBAR<br>1: Respective output of Mux28 is enabled to drive the TRIP9 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-53. 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<br>0: Respective output of Mux27 is disabled to drive the TRIP9 of EPWM-XBAR<br>1: Respective output of Mux27 is enabled to drive the TRIP9 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 26 | MUX26 | R/W | 0h | Selects the output of Mux26 to drive TRIP9 of EPWM-XBAR<br>0: Respective output of Mux26 is disabled to drive the TRIP9 of EPWM-XBAR<br>1: Respective output of Mux26 is enabled to drive the TRIP9 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25 | MUX25 | R/W | 0h | Selects the output of Mux25 to drive TRIP9 of EPWM-XBAR<br>0: Respective output of Mux25 is disabled to drive the TRIP9 of EPWM-XBAR<br>1: Respective output of Mux25 is enabled to drive the TRIP9 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 24 | MUX24 | R/W | 0h | Selects the output of Mux24 to drive TRIP9 of EPWM-XBAR<br>0: Respective output of Mux24 is disabled to drive the TRIP9 of EPWM-XBAR<br>1: Respective output of Mux24 is enabled to drive the TRIP9 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23 | MUX23 | R/W | 0h | Selects the output of Mux23 to drive TRIP9 of EPWM-XBAR<br>0: Respective output of Mux23 is disabled to drive the TRIP9 of EPWM-XBAR<br>1: Respective output of Mux23 is enabled to drive the TRIP9 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 22 | MUX22 | R/W | 0h | Selects the output of Mux22 to drive TRIP9 of EPWM-XBAR<br>0: Respective output of Mux22 is disabled to drive the TRIP9 of EPWM-XBAR<br>1: Respective output of Mux22 is enabled to drive the TRIP9 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21 | MUX21 | R/W | 0h | Selects the output of Mux21 to drive TRIP9 of EPWM-XBAR<br>0: Respective output of Mux21 is disabled to drive the TRIP9 of EPWM-XBAR<br>1: Respective output of Mux21 is enabled to drive the TRIP9 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 20 | MUX20 | R/W | 0h | Selects the output of Mux20 to drive TRIP9 of EPWM-XBAR<br>0: Respective output of Mux20 is disabled to drive the TRIP9 of EPWM-XBAR<br>1: Respective output of Mux20 is enabled to drive the TRIP9 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-53. 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<br>0: Respective output of Mux19 is disabled to drive the TRIP9 of EPWM-XBAR<br>1: Respective output of Mux19 is enabled to drive the TRIP9 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 18 | MUX18 | R/W | 0h | Selects the output of Mux18 to drive TRIP9 of EPWM-XBAR<br>0: Respective output of Mux18 is disabled to drive the TRIP9 of EPWM-XBAR<br>1: Respective output of Mux18 is enabled to drive the TRIP9 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17 | MUX17 | R/W | 0h | Selects the output of Mux17 to drive TRIP9 of EPWM-XBAR<br>0: Respective output of Mux17 is disabled to drive the TRIP9 of EPWM-XBAR<br>1: Respective output of Mux17 is enabled to drive the TRIP9 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 16 | MUX16 | R/W | 0h | Selects the output of Mux16 to drive TRIP9 of EPWM-XBAR<br>0: Respective output of Mux16 is disabled to drive the TRIP9 of EPWM-XBAR<br>1: Respective output of Mux16 is enabled to drive the TRIP9 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15 | MUX15 | R/W | 0h | Selects the output of Mux15 to drive TRIP9 of EPWM-XBAR<br>0: Respective output of Mux15 is disabled to drive the TRIP9 of EPWM-XBAR<br>1: Respective output of Mux15 is enabled to drive the TRIP9 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 14 | MUX14 | R/W | 0h | Selects the output of Mux14 to drive TRIP9 of EPWM-XBAR<br>0: Respective output of Mux14 is disabled to drive the TRIP9 of EPWM-XBAR<br>1: Respective output of Mux14 is enabled to drive the TRIP9 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13 | MUX13 | R/W | 0h | Selects the output of Mux13 to drive TRIP9 of EPWM-XBAR<br>0: Respective output of Mux13 is disabled to drive the TRIP9 of EPWM-XBAR<br>1: Respective output of Mux13 is enabled to drive the TRIP9 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 12 | MUX12 | R/W | 0h | Selects the output of Mux12 to drive TRIP9 of EPWM-XBAR<br>0: Respective output of Mux12 is disabled to drive the TRIP9 of EPWM-XBAR<br>1: Respective output of Mux12 is enabled to drive the TRIP9 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-53. 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<br>0: Respective output of Mux11 is disabled to drive the TRIP9 of EPWM-XBAR<br>1: Respective output of Mux11 is enabled to drive the TRIP9 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 10 | MUX10 | R/W | 0h | Selects the output of Mux10 to drive TRIP9 of EPWM-XBAR<br>0: Respective output of Mux10 is disabled to drive the TRIP9 of EPWM-XBAR<br>1: Respective output of Mux10 is enabled to drive the TRIP9 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9 | MUX9 | R/W | 0h | Selects the output of Mux9 to drive TRIP9 of EPWM-XBAR<br>0: Respective output of Mux9 is disabled to drive the TRIP9 of EPWM-XBAR<br>1: Respective output of Mux9 is enabled to drive the TRIP9 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 8 | MUX8 | R/W | 0h | Selects the output of Mux8 to drive TRIP9 of EPWM-XBAR<br>0: Respective output of Mux8 is disabled to drive the TRIP9 of EPWM-XBAR<br>1: Respective output of Mux8 is enabled to drive the TRIP9 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7 | MUX7 | R/W | 0h | Selects the output of Mux7 to drive TRIP9 of EPWM-XBAR<br>0: Respective output of Mux7 is disabled to drive the TRIP9 of EPWM-XBAR<br>1: Respective output of Mux7 is enabled to drive the TRIP9 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 6 | MUX6 | R/W | 0h | Selects the output of Mux6 to drive TRIP9 of EPWM-XBAR<br>0: Respective output of Mux6 is disabled to drive the TRIP9 of EPWM-XBAR<br>1: Respective output of Mux6 is enabled to drive the TRIP9 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5 | MUX5 | R/W | 0h | Selects the output of Mux5 to drive TRIP9 of EPWM-XBAR<br>0: Respective output of Mux5 is disabled to drive the TRIP9 of EPWM-XBAR<br>1: Respective output of Mux5 is enabled to drive the TRIP9 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 4 | MUX4 | R/W | 0h | Selects the output of Mux4 to drive TRIP9 of EPWM-XBAR<br>0: Respective output of Mux4 is disabled to drive the TRIP9 of EPWM-XBAR<br>1: Respective output of Mux4 is enabled to drive the TRIP9 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-53. 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<br>0: Respective output of Mux3 is disabled to drive the TRIP9 of EPWM-XBAR<br>1: Respective output of Mux3 is enabled to drive the TRIP9 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 2 | MUX2 | R/W | 0h | Selects the output of Mux2 to drive TRIP9 of EPWM-XBAR<br>0: Respective output of Mux2 is disabled to drive the TRIP9 of EPWM-XBAR<br>1: Respective output of Mux2 is enabled to drive the TRIP9 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1 | MUX1 | R/W | 0h | Selects the output of Mux1 to drive TRIP9 of EPWM-XBAR<br>0: Respective output of Mux1 is disabled to drive the TRIP9 of EPWM-XBAR<br>1: Respective output of Mux1 is enabled to drive the TRIP9 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 0 | MUX0 | R/W | 0h | Selects the output of mux0 to drive TRIP9 of EPWM-XBAR<br>0: Respective output of Mux0 is disabled to drive the TRIP9 of EPWM-XBAR<br>1: Respective output of Mux0 is enabled to drive the TRIP9 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.4.22 TRIP10MUXENABLE Register (Offset = 2Ah) [Reset = 00000000h]

TRIP10MUXENABLE is shown in Figure 9-49 and described in Table 9-54.

Return to the Summary Table.

ePWM XBAR Mux Enable for TRIP10

**Figure 9-49. TRIP10MUXENABLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MUX31 | MUX30 | MUX29 | MUX28 | MUX27 | MUX26 | MUX25 | MUX24 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MUX23 | MUX22 | MUX21 | MUX20 | MUX19 | MUX18 | MUX17 | MUX16 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MUX15 | MUX14 | MUX13 | MUX12 | MUX11 | MUX10 | MUX9 | MUX8 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | MUX6 | MUX5 | MUX4 | MUX3 | MUX2 | MUX1 | MUX0 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 9-54. 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<br>0: Respective output of Mux31 is disabled to drive the TRIP10 of EPWM-XBAR<br>1: Respective output of Mux31 is enabled to drive the TRIP10 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 30 | MUX30 | R/W | 0h | Selects the output of Mux30 to drive TRIP10 of EPWM-XBAR<br>0: Respective output of Mux30 is disabled to drive the TRIP10 of EPWM-XBAR<br>1: Respective output of Mux30 is enabled to drive the TRIP10 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29 | MUX29 | R/W | 0h | Selects the output of Mux29 to drive TRIP10 of EPWM-XBAR<br>0: Respective output of Mux29 is disabled to drive the TRIP10 of EPWM-XBAR<br>1: Respective output of Mux29 is enabled to drive the TRIP10 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 28 | MUX28 | R/W | 0h | Selects the output of Mux28 to drive TRIP10 of EPWM-XBAR<br>0: Respective output of Mux28 is disabled to drive the TRIP10 of EPWM-XBAR<br>1: Respective output of Mux28 is enabled to drive the TRIP10 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-54. 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<br>0: Respective output of Mux27 is disabled to drive the TRIP10 of EPWM-XBAR<br>1: Respective output of Mux27 is enabled to drive the TRIP10 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 26 | MUX26 | R/W | 0h | Selects the output of Mux26 to drive TRIP10 of EPWM-XBAR<br>0: Respective output of Mux26 is disabled to drive the TRIP10 of EPWM-XBAR<br>1: Respective output of Mux26 is enabled to drive the TRIP10 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25 | MUX25 | R/W | 0h | Selects the output of Mux25 to drive TRIP10 of EPWM-XBAR<br>0: Respective output of Mux25 is disabled to drive the TRIP10 of EPWM-XBAR<br>1: Respective output of Mux25 is enabled to drive the TRIP10 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 24 | MUX24 | R/W | 0h | Selects the output of Mux24 to drive TRIP10 of EPWM-XBAR<br>0: Respective output of Mux24 is disabled to drive the TRIP10 of EPWM-XBAR<br>1: Respective output of Mux24 is enabled to drive the TRIP10 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23 | MUX23 | R/W | 0h | Selects the output of Mux23 to drive TRIP10 of EPWM-XBAR<br>0: Respective output of Mux23 is disabled to drive the TRIP10 of EPWM-XBAR<br>1: Respective output of Mux23 is enabled to drive the TRIP10 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 22 | MUX22 | R/W | 0h | Selects the output of Mux22 to drive TRIP10 of EPWM-XBAR<br>0: Respective output of Mux22 is disabled to drive the TRIP10 of EPWM-XBAR<br>1: Respective output of Mux22 is enabled to drive the TRIP10 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21 | MUX21 | R/W | 0h | Selects the output of Mux21 to drive TRIP10 of EPWM-XBAR<br>0: Respective output of Mux21 is disabled to drive the TRIP10 of EPWM-XBAR<br>1: Respective output of Mux21 is enabled to drive the TRIP10 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 20 | MUX20 | R/W | 0h | Selects the output of Mux20 to drive TRIP10 of EPWM-XBAR<br>0: Respective output of Mux20 is disabled to drive the TRIP10 of EPWM-XBAR<br>1: Respective output of Mux20 is enabled to drive the TRIP10 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-54. 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<br>0: Respective output of Mux19 is disabled to drive the TRIP10 of EPWM-XBAR<br>1: Respective output of Mux19 is enabled to drive the TRIP10 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 18 | MUX18 | R/W | 0h | Selects the output of Mux18 to drive TRIP10 of EPWM-XBAR<br>0: Respective output of Mux18 is disabled to drive the TRIP10 of EPWM-XBAR<br>1: Respective output of Mux18 is enabled to drive the TRIP10 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17 | MUX17 | R/W | 0h | Selects the output of Mux17 to drive TRIP10 of EPWM-XBAR<br>0: Respective output of Mux17 is disabled to drive the TRIP10 of EPWM-XBAR<br>1: Respective output of Mux17 is enabled to drive the TRIP10 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 16 | MUX16 | R/W | 0h | Selects the output of Mux16 to drive TRIP10 of EPWM-XBAR<br>0: Respective output of Mux16 is disabled to drive the TRIP10 of EPWM-XBAR<br>1: Respective output of Mux16 is enabled to drive the TRIP10 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15 | MUX15 | R/W | 0h | Selects the output of Mux15 to drive TRIP10 of EPWM-XBAR<br>0: Respective output of Mux15 is disabled to drive the TRIP10 of EPWM-XBAR<br>1: Respective output of Mux15 is enabled to drive the TRIP10 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 14 | MUX14 | R/W | 0h | Selects the output of Mux14 to drive TRIP10 of EPWM-XBAR<br>0: Respective output of Mux14 is disabled to drive the TRIP10 of EPWM-XBAR<br>1: Respective output of Mux14 is enabled to drive the TRIP10 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13 | MUX13 | R/W | 0h | Selects the output of Mux13 to drive TRIP10 of EPWM-XBAR<br>0: Respective output of Mux13 is disabled to drive the TRIP10 of EPWM-XBAR<br>1: Respective output of Mux13 is enabled to drive the TRIP10 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 12 | MUX12 | R/W | 0h | Selects the output of Mux12 to drive TRIP10 of EPWM-XBAR<br>0: Respective output of Mux12 is disabled to drive the TRIP10 of EPWM-XBAR<br>1: Respective output of Mux12 is enabled to drive the TRIP10 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-54. 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<br>0: Respective output of Mux11 is disabled to drive the TRIP10 of EPWM-XBAR<br>1: Respective output of Mux11 is enabled to drive the TRIP10 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 10 | MUX10 | R/W | 0h | Selects the output of Mux10 to drive TRIP10 of EPWM-XBAR<br>0: Respective output of Mux10 is disabled to drive the TRIP10 of EPWM-XBAR<br>1: Respective output of Mux10 is enabled to drive the TRIP10 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9 | MUX9 | R/W | 0h | Selects the output of Mux9 to drive TRIP10 of EPWM-XBAR<br>0: Respective output of Mux9 is disabled to drive the TRIP10 of EPWM-XBAR<br>1: Respective output of Mux9 is enabled to drive the TRIP10 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 8 | MUX8 | R/W | 0h | Selects the output of Mux8 to drive TRIP10 of EPWM-XBAR<br>0: Respective output of Mux8 is disabled to drive the TRIP10 of EPWM-XBAR<br>1: Respective output of Mux8 is enabled to drive the TRIP10 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7 | MUX7 | R/W | 0h | Selects the output of Mux7 to drive TRIP10 of EPWM-XBAR<br>0: Respective output of Mux7 is disabled to drive the TRIP10 of EPWM-XBAR<br>1: Respective output of Mux7 is enabled to drive the TRIP10 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 6 | MUX6 | R/W | 0h | Selects the output of Mux6 to drive TRIP10 of EPWM-XBAR<br>0: Respective output of Mux6 is disabled to drive the TRIP10 of EPWM-XBAR<br>1: Respective output of Mux6 is enabled to drive the TRIP10 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5 | MUX5 | R/W | 0h | Selects the output of Mux5 to drive TRIP10 of EPWM-XBAR<br>0: Respective output of Mux5 is disabled to drive the TRIP10 of EPWM-XBAR<br>1: Respective output of Mux5 is enabled to drive the TRIP10 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 4 | MUX4 | R/W | 0h | Selects the output of Mux4 to drive TRIP10 of EPWM-XBAR<br>0: Respective output of Mux4 is disabled to drive the TRIP10 of EPWM-XBAR<br>1: Respective output of Mux4 is enabled to drive the TRIP10 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-54. 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<br>0: Respective output of Mux3 is disabled to drive the TRIP10 of EPWM-XBAR<br>1: Respective output of Mux3 is enabled to drive the TRIP10 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 2 | MUX2 | R/W | 0h | Selects the output of Mux2 to drive TRIP10 of EPWM-XBAR<br>0: Respective output of Mux2 is disabled to drive the TRIP10 of EPWM-XBAR<br>1: Respective output of Mux2 is enabled to drive the TRIP10 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1 | MUX1 | R/W | 0h | Selects the output of Mux1 to drive TRIP10 of EPWM-XBAR<br>0: Respective output of Mux1 is disabled to drive the TRIP10 of EPWM-XBAR<br>1: Respective output of Mux1 is enabled to drive the TRIP10 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 0 | MUX0 | R/W | 0h | Selects the output of mux0 to drive TRIP10 of EPWM-XBAR<br>0: Respective output of Mux0 is disabled to drive the TRIP10 of EPWM-XBAR<br>1: Respective output of Mux0 is enabled to drive the TRIP10 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.4.23 TRIP11MUXENABLE Register (Offset = 2Ch) [Reset = 00000000h]

TRIP11MUXENABLE is shown in Figure 9-50 and described in Table 9-55.

Return to the Summary Table.

ePWM XBAR Mux Enable for TRIP11

**Figure 9-50. TRIP11MUXENABLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MUX31 | MUX30 | MUX29 | MUX28 | MUX27 | MUX26 | MUX25 | MUX24 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MUX23 | MUX22 | MUX21 | MUX20 | MUX19 | MUX18 | MUX17 | MUX16 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MUX15 | MUX14 | MUX13 | MUX12 | MUX11 | MUX10 | MUX9 | MUX8 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | MUX6 | MUX5 | MUX4 | MUX3 | MUX2 | MUX1 | MUX0 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 9-55. 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<br>0: Respective output of Mux31 is disabled to drive the TRIP11 of EPWM-XBAR<br>1: Respective output of Mux31 is enabled to drive the TRIP11 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 30 | MUX30 | R/W | 0h | Selects the output of Mux30 to drive TRIP11 of EPWM-XBAR<br>0: Respective output of Mux30 is disabled to drive the TRIP11 of EPWM-XBAR<br>1: Respective output of Mux30 is enabled to drive the TRIP11 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29 | MUX29 | R/W | 0h | Selects the output of Mux29 to drive TRIP11 of EPWM-XBAR<br>0: Respective output of Mux29 is disabled to drive the TRIP11 of EPWM-XBAR<br>1: Respective output of Mux29 is enabled to drive the TRIP11 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 28 | MUX28 | R/W | 0h | Selects the output of Mux28 to drive TRIP11 of EPWM-XBAR<br>0: Respective output of Mux28 is disabled to drive the TRIP11 of EPWM-XBAR<br>1: Respective output of Mux28 is enabled to drive the TRIP11 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-55. 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<br>0: Respective output of Mux27 is disabled to drive the TRIP11 of EPWM-XBAR<br>1: Respective output of Mux27 is enabled to drive the TRIP11 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 26 | MUX26 | R/W | 0h | Selects the output of Mux26 to drive TRIP11 of EPWM-XBAR<br>0: Respective output of Mux26 is disabled to drive the TRIP11 of EPWM-XBAR<br>1: Respective output of Mux26 is enabled to drive the TRIP11 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25 | MUX25 | R/W | 0h | Selects the output of Mux25 to drive TRIP11 of EPWM-XBAR<br>0: Respective output of Mux25 is disabled to drive the TRIP11 of EPWM-XBAR<br>1: Respective output of Mux25 is enabled to drive the TRIP11 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 24 | MUX24 | R/W | 0h | Selects the output of Mux24 to drive TRIP11 of EPWM-XBAR<br>0: Respective output of Mux24 is disabled to drive the TRIP11 of EPWM-XBAR<br>1: Respective output of Mux24 is enabled to drive the TRIP11 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23 | MUX23 | R/W | 0h | Selects the output of Mux23 to drive TRIP11 of EPWM-XBAR<br>0: Respective output of Mux23 is disabled to drive the TRIP11 of EPWM-XBAR<br>1: Respective output of Mux23 is enabled to drive the TRIP11 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 22 | MUX22 | R/W | 0h | Selects the output of Mux22 to drive TRIP11 of EPWM-XBAR<br>0: Respective output of Mux22 is disabled to drive the TRIP11 of EPWM-XBAR<br>1: Respective output of Mux22 is enabled to drive the TRIP11 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21 | MUX21 | R/W | 0h | Selects the output of Mux21 to drive TRIP11 of EPWM-XBAR<br>0: Respective output of Mux21 is disabled to drive the TRIP11 of EPWM-XBAR<br>1: Respective output of Mux21 is enabled to drive the TRIP11 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 20 | MUX20 | R/W | 0h | Selects the output of Mux20 to drive TRIP11 of EPWM-XBAR<br>0: Respective output of Mux20 is disabled to drive the TRIP11 of EPWM-XBAR<br>1: Respective output of Mux20 is enabled to drive the TRIP11 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-55. 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<br>0: Respective output of Mux19 is disabled to drive the TRIP11 of EPWM-XBAR<br>1: Respective output of Mux19 is enabled to drive the TRIP11 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 18 | MUX18 | R/W | 0h | Selects the output of Mux18 to drive TRIP11 of EPWM-XBAR<br>0: Respective output of Mux18 is disabled to drive the TRIP11 of EPWM-XBAR<br>1: Respective output of Mux18 is enabled to drive the TRIP11 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17 | MUX17 | R/W | 0h | Selects the output of Mux17 to drive TRIP11 of EPWM-XBAR<br>0: Respective output of Mux17 is disabled to drive the TRIP11 of EPWM-XBAR<br>1: Respective output of Mux17 is enabled to drive the TRIP11 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 16 | MUX16 | R/W | 0h | Selects the output of Mux16 to drive TRIP11 of EPWM-XBAR<br>0: Respective output of Mux16 is disabled to drive the TRIP11 of EPWM-XBAR<br>1: Respective output of Mux16 is enabled to drive the TRIP11 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15 | MUX15 | R/W | 0h | Selects the output of Mux15 to drive TRIP11 of EPWM-XBAR<br>0: Respective output of Mux15 is disabled to drive the TRIP11 of EPWM-XBAR<br>1: Respective output of Mux15 is enabled to drive the TRIP11 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 14 | MUX14 | R/W | 0h | Selects the output of Mux14 to drive TRIP11 of EPWM-XBAR<br>0: Respective output of Mux14 is disabled to drive the TRIP11 of EPWM-XBAR<br>1: Respective output of Mux14 is enabled to drive the TRIP11 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13 | MUX13 | R/W | 0h | Selects the output of Mux13 to drive TRIP11 of EPWM-XBAR<br>0: Respective output of Mux13 is disabled to drive the TRIP11 of EPWM-XBAR<br>1: Respective output of Mux13 is enabled to drive the TRIP11 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 12 | MUX12 | R/W | 0h | Selects the output of Mux12 to drive TRIP11 of EPWM-XBAR<br>0: Respective output of Mux12 is disabled to drive the TRIP11 of EPWM-XBAR<br>1: Respective output of Mux12 is enabled to drive the TRIP11 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-55. 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<br>0: Respective output of Mux11 is disabled to drive the TRIP11 of EPWM-XBAR<br>1: Respective output of Mux11 is enabled to drive the TRIP11 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 10 | MUX10 | R/W | 0h | Selects the output of Mux10 to drive TRIP11 of EPWM-XBAR<br>0: Respective output of Mux10 is disabled to drive the TRIP11 of EPWM-XBAR<br>1: Respective output of Mux10 is enabled to drive the TRIP11 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9 | MUX9 | R/W | 0h | Selects the output of Mux9 to drive TRIP11 of EPWM-XBAR<br>0: Respective output of Mux9 is disabled to drive the TRIP11 of EPWM-XBAR<br>1: Respective output of Mux9 is enabled to drive the TRIP11 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 8 | MUX8 | R/W | 0h | Selects the output of Mux8 to drive TRIP11 of EPWM-XBAR<br>0: Respective output of Mux8 is disabled to drive the TRIP11 of EPWM-XBAR<br>1: Respective output of Mux8 is enabled to drive the TRIP11 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7 | MUX7 | R/W | 0h | Selects the output of Mux7 to drive TRIP11 of EPWM-XBAR<br>0: Respective output of Mux7 is disabled to drive the TRIP11 of EPWM-XBAR<br>1: Respective output of Mux7 is enabled to drive the TRIP11 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 6 | MUX6 | R/W | 0h | Selects the output of Mux6 to drive TRIP11 of EPWM-XBAR<br>0: Respective output of Mux6 is disabled to drive the TRIP11 of EPWM-XBAR<br>1: Respective output of Mux6 is enabled to drive the TRIP11 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5 | MUX5 | R/W | 0h | Selects the output of Mux5 to drive TRIP11 of EPWM-XBAR<br>0: Respective output of Mux5 is disabled to drive the TRIP11 of EPWM-XBAR<br>1: Respective output of Mux5 is enabled to drive the TRIP11 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 4 | MUX4 | R/W | 0h | Selects the output of Mux4 to drive TRIP11 of EPWM-XBAR<br>0: Respective output of Mux4 is disabled to drive the TRIP11 of EPWM-XBAR<br>1: Respective output of Mux4 is enabled to drive the TRIP11 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-55. 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<br>0: Respective output of Mux3 is disabled to drive the TRIP11 of EPWM-XBAR<br>1: Respective output of Mux3 is enabled to drive the TRIP11 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 2 | MUX2 | R/W | 0h | Selects the output of Mux2 to drive TRIP11 of EPWM-XBAR<br>0: Respective output of Mux2 is disabled to drive the TRIP11 of EPWM-XBAR<br>1: Respective output of Mux2 is enabled to drive the TRIP11 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1 | MUX1 | R/W | 0h | Selects the output of Mux1 to drive TRIP11 of EPWM-XBAR<br>0: Respective output of Mux1 is disabled to drive the TRIP11 of EPWM-XBAR<br>1: Respective output of Mux1 is enabled to drive the TRIP11 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 0 | MUX0 | R/W | 0h | Selects the output of mux0 to drive TRIP11 of EPWM-XBAR<br>0: Respective output of Mux0 is disabled to drive the TRIP11 of EPWM-XBAR<br>1: Respective output of Mux0 is enabled to drive the TRIP11 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.4.24 TRIP12MUXENABLE Register (Offset = 2Eh) [Reset = 00000000h]

TRIP12MUXENABLE is shown in Figure 9-51 and described in Table 9-56.

Return to the Summary Table.

ePWM XBAR Mux Enable for TRIP12

**Figure 9-51. TRIP12MUXENABLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MUX31 | MUX30 | MUX29 | MUX28 | MUX27 | MUX26 | MUX25 | MUX24 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MUX23 | MUX22 | MUX21 | MUX20 | MUX19 | MUX18 | MUX17 | MUX16 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MUX15 | MUX14 | MUX13 | MUX12 | MUX11 | MUX10 | MUX9 | MUX8 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | MUX6 | MUX5 | MUX4 | MUX3 | MUX2 | MUX1 | MUX0 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 9-56. 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<br>0: Respective output of Mux31 is disabled to drive the TRIP12 of EPWM-XBAR<br>1: Respective output of Mux31 is enabled to drive the TRIP12 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 30 | MUX30 | R/W | 0h | Selects the output of Mux30 to drive TRIP12 of EPWM-XBAR<br>0: Respective output of Mux30 is disabled to drive the TRIP12 of EPWM-XBAR<br>1: Respective output of Mux30 is enabled to drive the TRIP12 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29 | MUX29 | R/W | 0h | Selects the output of Mux29 to drive TRIP12 of EPWM-XBAR<br>0: Respective output of Mux29 is disabled to drive the TRIP12 of EPWM-XBAR<br>1: Respective output of Mux29 is enabled to drive the TRIP12 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 28 | MUX28 | R/W | 0h | Selects the output of Mux28 to drive TRIP12 of EPWM-XBAR<br>0: Respective output of Mux28 is disabled to drive the TRIP12 of EPWM-XBAR<br>1: Respective output of Mux28 is enabled to drive the TRIP12 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-56. 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<br>0: Respective output of Mux27 is disabled to drive the TRIP12 of EPWM-XBAR<br>1: Respective output of Mux27 is enabled to drive the TRIP12 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 26 | MUX26 | R/W | 0h | Selects the output of Mux26 to drive TRIP12 of EPWM-XBAR<br>0: Respective output of Mux26 is disabled to drive the TRIP12 of EPWM-XBAR<br>1: Respective output of Mux26 is enabled to drive the TRIP12 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25 | MUX25 | R/W | 0h | Selects the output of Mux25 to drive TRIP12 of EPWM-XBAR<br>0: Respective output of Mux25 is disabled to drive the TRIP12 of EPWM-XBAR<br>1: Respective output of Mux25 is enabled to drive the TRIP12 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 24 | MUX24 | R/W | 0h | Selects the output of Mux24 to drive TRIP12 of EPWM-XBAR<br>0: Respective output of Mux24 is disabled to drive the TRIP12 of EPWM-XBAR<br>1: Respective output of Mux24 is enabled to drive the TRIP12 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23 | MUX23 | R/W | 0h | Selects the output of Mux23 to drive TRIP12 of EPWM-XBAR<br>0: Respective output of Mux23 is disabled to drive the TRIP12 of EPWM-XBAR<br>1: Respective output of Mux23 is enabled to drive the TRIP12 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 22 | MUX22 | R/W | 0h | Selects the output of Mux22 to drive TRIP12 of EPWM-XBAR<br>0: Respective output of Mux22 is disabled to drive the TRIP12 of EPWM-XBAR<br>1: Respective output of Mux22 is enabled to drive the TRIP12 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21 | MUX21 | R/W | 0h | Selects the output of Mux21 to drive TRIP12 of EPWM-XBAR<br>0: Respective output of Mux21 is disabled to drive the TRIP12 of EPWM-XBAR<br>1: Respective output of Mux21 is enabled to drive the TRIP12 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 20 | MUX20 | R/W | 0h | Selects the output of Mux20 to drive TRIP12 of EPWM-XBAR<br>0: Respective output of Mux20 is disabled to drive the TRIP12 of EPWM-XBAR<br>1: Respective output of Mux20 is enabled to drive the TRIP12 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-56. 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<br>0: Respective output of Mux19 is disabled to drive the TRIP12 of EPWM-XBAR<br>1: Respective output of Mux19 is enabled to drive the TRIP12 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 18 | MUX18 | R/W | 0h | Selects the output of Mux18 to drive TRIP12 of EPWM-XBAR<br>0: Respective output of Mux18 is disabled to drive the TRIP12 of EPWM-XBAR<br>1: Respective output of Mux18 is enabled to drive the TRIP12 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17 | MUX17 | R/W | 0h | Selects the output of Mux17 to drive TRIP12 of EPWM-XBAR<br>0: Respective output of Mux17 is disabled to drive the TRIP12 of EPWM-XBAR<br>1: Respective output of Mux17 is enabled to drive the TRIP12 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 16 | MUX16 | R/W | 0h | Selects the output of Mux16 to drive TRIP12 of EPWM-XBAR<br>0: Respective output of Mux16 is disabled to drive the TRIP12 of EPWM-XBAR<br>1: Respective output of Mux16 is enabled to drive the TRIP12 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15 | MUX15 | R/W | 0h | Selects the output of Mux15 to drive TRIP12 of EPWM-XBAR<br>0: Respective output of Mux15 is disabled to drive the TRIP12 of EPWM-XBAR<br>1: Respective output of Mux15 is enabled to drive the TRIP12 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 14 | MUX14 | R/W | 0h | Selects the output of Mux14 to drive TRIP12 of EPWM-XBAR<br>0: Respective output of Mux14 is disabled to drive the TRIP12 of EPWM-XBAR<br>1: Respective output of Mux14 is enabled to drive the TRIP12 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13 | MUX13 | R/W | 0h | Selects the output of Mux13 to drive TRIP12 of EPWM-XBAR<br>0: Respective output of Mux13 is disabled to drive the TRIP12 of EPWM-XBAR<br>1: Respective output of Mux13 is enabled to drive the TRIP12 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 12 | MUX12 | R/W | 0h | Selects the output of Mux12 to drive TRIP12 of EPWM-XBAR<br>0: Respective output of Mux12 is disabled to drive the TRIP12 of EPWM-XBAR<br>1: Respective output of Mux12 is enabled to drive the TRIP12 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-56. 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<br>0: Respective output of Mux11 is disabled to drive the TRIP12 of EPWM-XBAR<br>1: Respective output of Mux11 is enabled to drive the TRIP12 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 10 | MUX10 | R/W | 0h | Selects the output of Mux10 to drive TRIP12 of EPWM-XBAR<br>0: Respective output of Mux10 is disabled to drive the TRIP12 of EPWM-XBAR<br>1: Respective output of Mux10 is enabled to drive the TRIP12 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9 | MUX9 | R/W | 0h | Selects the output of Mux9 to drive TRIP12 of EPWM-XBAR<br>0: Respective output of Mux9 is disabled to drive the TRIP12 of EPWM-XBAR<br>1: Respective output of Mux9 is enabled to drive the TRIP12 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 8 | MUX8 | R/W | 0h | Selects the output of Mux8 to drive TRIP12 of EPWM-XBAR<br>0: Respective output of Mux8 is disabled to drive the TRIP12 of EPWM-XBAR<br>1: Respective output of Mux8 is enabled to drive the TRIP12 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7 | MUX7 | R/W | 0h | Selects the output of Mux7 to drive TRIP12 of EPWM-XBAR<br>0: Respective output of Mux7 is disabled to drive the TRIP12 of EPWM-XBAR<br>1: Respective output of Mux7 is enabled to drive the TRIP12 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 6 | MUX6 | R/W | 0h | Selects the output of Mux6 to drive TRIP12 of EPWM-XBAR<br>0: Respective output of Mux6 is disabled to drive the TRIP12 of EPWM-XBAR<br>1: Respective output of Mux6 is enabled to drive the TRIP12 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5 | MUX5 | R/W | 0h | Selects the output of Mux5 to drive TRIP12 of EPWM-XBAR<br>0: Respective output of Mux5 is disabled to drive the TRIP12 of EPWM-XBAR<br>1: Respective output of Mux5 is enabled to drive the TRIP12 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 4 | MUX4 | R/W | 0h | Selects the output of Mux4 to drive TRIP12 of EPWM-XBAR<br>0: Respective output of Mux4 is disabled to drive the TRIP12 of EPWM-XBAR<br>1: Respective output of Mux4 is enabled to drive the TRIP12 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-56. 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<br>0: Respective output of Mux3 is disabled to drive the TRIP12 of EPWM-XBAR<br>1: Respective output of Mux3 is enabled to drive the TRIP12 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 2 | MUX2 | R/W | 0h | Selects the output of Mux2 to drive TRIP12 of EPWM-XBAR<br>0: Respective output of Mux2 is disabled to drive the TRIP12 of EPWM-XBAR<br>1: Respective output of Mux2 is enabled to drive the TRIP12 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1 | MUX1 | R/W | 0h | Selects the output of Mux1 to drive TRIP12 of EPWM-XBAR<br>0: Respective output of Mux1 is disabled to drive the TRIP12 of EPWM-XBAR<br>1: Respective output of Mux1 is enabled to drive the TRIP12 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 0 | MUX0 | R/W | 0h | Selects the output of mux0 to drive TRIP12 of EPWM-XBAR<br>0: Respective output of Mux0 is disabled to drive the TRIP12 of EPWM-XBAR<br>1: Respective output of Mux0 is enabled to drive the TRIP12 of EPWM-XBAR<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.4.25 TRIPOUTINV Register (Offset = 38h) [Reset = 00000000h]

TRIPOUTINV is shown in Figure 9-52 and described in Table 9-57.

Return to the Summary Table.

ePWM XBAR Output Inversion Register

**Figure 9-52. 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 9-57. 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<br>0: drives active high output<br>1: drives active-low output<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 6 | TRIP11 | R/W | 0h | Selects polarity for TRIP11 of EPWM-XBAR<br>0: drives active high output<br>1: drives active-low output<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5 | TRIP10 | R/W | 0h | Selects polarity for TRIP10 of EPWM-XBAR<br>0: drives active high output<br>1: drives active-low output<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 4 | TRIP9 | R/W | 0h | Selects polarity for TRIP9 of EPWM-XBAR<br>0: drives active high output<br>1: drives active-low output<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 3 | TRIP8 | R/W | 0h | Selects polarity for TRIP8 of EPWM-XBAR<br>0: drives active high output<br>1: drives active-low output<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 2 | TRIP7 | R/W | 0h | Selects polarity for TRIP7 of EPWM-XBAR<br>0: drives active high output<br>1: drives active-low output<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-57. TRIPOUTINV Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1 | TRIP5 | R/W | 0h | Selects polarity for TRIP5 of EPWM-XBAR<br>0: drives active high output<br>1: drives active-low output<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 0 | TRIP4 | R/W | 0h | Selects polarity for TRIP4 of EPWM-XBAR<br>0: drives active high output<br>1: drives active-low output<br>Refer to the EPWM X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.4.26 TRIPLOCK Register (Offset = 3Eh) [Reset = 00000000h]

TRIPLOCK is shown in Figure 9-53 and described in Table 9-58.

Return to the Summary Table.

ePWM XBAR Configuration Lock register

**Figure 9-53. 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/WSonce-0h |

**Table 9-58. 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<br>Reset type: CPU1.SYSRSn |
| 15-1 | RESERVED | R-0 | 0h | Reserved |
| 0 | LOCK | R/WSonce | 0h | Locks the configuration for EPWM-XBAR. Once the configuration is locked, writes to the below registers for EPWM-XBAR is blocked.<br>Registers Affected by the LOCK mechanism:<br>EPWM-XBAROUTyMUX0TO15CFG<br>EPWM-XBAROUTyMUX16TO31CFG<br>EPWM-XBAROUTyMUXENABLE<br>EPWM-XBAROUTLATEN<br>EPWM-XBAROUTINV<br>0: Writes to the above registers are allowed<br>1: Writes to the above registers are blocked<br>Note:<br>[1] LOCK mechanism only apples to writes. Reads are never blocked.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.5 CLB_XBAR_REGS Registers

Table 9-59 lists the memory-mapped registers for the CLB_XBAR_REGS registers. All register offset addresses not listed in Table 9-59 should be considered as reserved locations and the register contents should not be modified.

**Table 9-59. CLB_XBAR_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | AUXSIG0MUX0TO15CFG | CLB XBAR Mux Configuration for Output-0 | EALLOW | Go |
| 2h | AUXSIG0MUX16TO31CFG | CLB XBAR Mux Configuration for Output-0 | EALLOW | Go |
| 4h | AUXSIG1MUX0TO15CFG | CLB XBAR Mux Configuration for Output-1 | EALLOW | Go |
| 6h | AUXSIG1MUX16TO31CFG | CLB XBAR Mux Configuration for Output-1 | EALLOW | Go |
| 8h | AUXSIG2MUX0TO15CFG | CLB XBAR Mux Configuration for Output-2 | EALLOW | Go |
| Ah | AUXSIG2MUX16TO31CFG | CLB XBAR Mux Configuration for Output-2 | EALLOW | Go |
| Ch | AUXSIG3MUX0TO15CFG | CLB XBAR Mux Configuration for Output-3 | EALLOW | Go |
| Eh | AUXSIG3MUX16TO31CFG | CLB XBAR Mux Configuration for Output-3 | EALLOW | Go |
| 10h | AUXSIG4MUX0TO15CFG | CLB XBAR Mux Configuration for Output-4 | EALLOW | Go |
| 12h | AUXSIG4MUX16TO31CFG | CLB XBAR Mux Configuration for Output-4 | EALLOW | Go |
| 14h | AUXSIG5MUX0TO15CFG | CLB XBAR Mux Configuration for Output-5 | EALLOW | Go |
| 16h | AUXSIG5MUX16TO31CFG | CLB XBAR Mux Configuration for Output-5 | EALLOW | Go |
| 18h | AUXSIG6MUX0TO15CFG | CLB XBAR Mux Configuration for Output-6 | EALLOW | Go |
| 1Ah | AUXSIG6MUX16TO31CFG | CLB XBAR Mux Configuration for Output-6 | EALLOW | Go |
| 1Ch | AUXSIG7MUX0TO15CFG | CLB XBAR Mux Configuration for Output-7 | EALLOW | Go |
| 1Eh | AUXSIG7MUX16TO31CFG | CLB XBAR Mux Configuration for Output-7 | EALLOW | Go |
| 20h | AUXSIG0MUXENABLE | CLB XBAR Mux Enable Register for Output-0 | EALLOW | Go |
| 22h | AUXSIG1MUXENABLE | CLB XBAR Mux Enable Register for Output-1 | EALLOW | Go |
| 24h | AUXSIG2MUXENABLE | CLB XBAR Mux Enable Register for Output-2 | EALLOW | Go |
| 26h | AUXSIG3MUXENABLE | CLB XBAR Mux Enable Register for Output-3 | EALLOW | Go |
| 28h | AUXSIG4MUXENABLE | CLB XBAR Mux Enable Register for Output-4 | EALLOW | Go |
| 2Ah | AUXSIG5MUXENABLE | CLB XBAR Mux Enable Register for Output-5 | EALLOW | Go |
| 2Ch | AUXSIG6MUXENABLE | CLB XBAR Mux Enable Register for Output-6 | EALLOW | Go |
| 2Eh | AUXSIG7MUXENABLE | CLB XBAR Mux Enable Register for Output-7 | EALLOW | Go |
| 38h | AUXSIGOUTINV | CLB XBAR Output Inversion Register | EALLOW | Go |
| 3Eh | AUXSIGLOCK | ClbXbar Configuration Lock register | EALLOW | Go |

Complex bit access types are encoded to fit into small table cells. Table 9-60 shows the codes that are used for access types in this section.

**Table 9-60. CLB_XBAR_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| R-0 | R<br>-0 | Read<br>Returns 0s |
| Write Type | | |
| W | W | Write |
| WSonce | W<br>Sonce | Write<br>Set once |
| Reset or Default Value | | |
| *-n* | | Value after reset or the default value |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-60. CLB_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. |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.5.1 AUXSIG0MUX0TO15CFG Register (Offset = 0h) [Reset = 00000000h]

AUXSIG0MUX0TO15CFG is shown in Figure 9-54 and described in Table 9-61.

Return to the Summary Table.

CLB XBAR Mux Configuration for Output-0

**Figure 9-54. AUXSIG0MUX0TO15CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX15 | | MUX14 | | MUX13 | | MUX12 | | MUX11 | | MUX10 | | MUX9 | | MUX8 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | | MUX6 | | MUX5 | | MUX4 | | MUX3 | | MUX2 | | MUX1 | | MUX0 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-61. AUXSIG0MUX0TO15CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX15 | R/W | 0h | Group Select Bits for MUX15: Selects 4X1 group output for AUXSIG0 of CLB-XBAR<br>00 : Select .0 input for MUX15<br>01 : Select .1 input for MUX15<br>10 : Select .2 input for MUX15<br>11 : Select .3 input for MUX15<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX14 | R/W | 0h | Group Select Bits for MUX14: Selects 4X1 group output for AUXSIG0 of CLB-XBAR<br>00 : Select .0 input for MUX14<br>01 : Select .1 input for MUX14<br>10 : Select .2 input for MUX14<br>11 : Select .3 input for MUX14<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX13 | R/W | 0h | Group Select Bits for MUX13: Selects 4X1 group output for AUXSIG0 of CLB-XBAR<br>00 : Select .0 input for MUX13<br>01 : Select .1 input for MUX13<br>10 : Select .2 input for MUX13<br>11 : Select .3 input for MUX13<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX12 | R/W | 0h | Group Select Bits for MUX12: Selects 4X1 group output for AUXSIG0 of CLB-XBAR<br>00 : Select .0 input for MUX12<br>01 : Select .1 input for MUX12<br>10 : Select .2 input for MUX12<br>11 : Select .3 input for MUX12<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX11 | R/W | 0h | Group Select Bits for MUX11: Selects 4X1 group output for AUXSIG0 of CLB-XBAR<br>00 : Select .0 input for MUX11<br>01 : Select .1 input for MUX11<br>10 : Select .2 input for MUX11<br>11 : Select .3 input for MUX11<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-61. AUXSIG0MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 21-20 | MUX10 | R/W | 0h | Group Select Bits for MUX10: Selects 4X1 group output for AUXSIG0 of CLB-XBAR<br>00 : Select .0 input for MUX10<br>01 : Select .1 input for MUX10<br>10 : Select .2 input for MUX10<br>11 : Select .3 input for MUX10<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 19-18 | MUX9 | R/W | 0h | Group Select Bits for MUX9: Selects 4X1 group output for AUXSIG0 of CLB-XBAR<br>00 : Select .0 input for MUX9<br>01 : Select .1 input for MUX9<br>10 : Select .2 input for MUX9<br>11 : Select .3 input for MUX9<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX8 | R/W | 0h | Group Select Bits for MUX8: Selects 4X1 group output for AUXSIG0 of CLB-XBAR<br>00 : Select .0 input for MUX8<br>01 : Select .1 input for MUX8<br>10 : Select .2 input for MUX8<br>11 : Select .3 input for MUX8<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX7 | R/W | 0h | Group Select Bits for MUX7: Selects 4X1 group output for AUXSIG0 of CLB-XBAR<br>00 : Select .0 input for MUX7<br>01 : Select .1 input for MUX7<br>10 : Select .2 input for MUX7<br>11 : Select .3 input for MUX7<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX6 | R/W | 0h | Group Select Bits for MUX6: Selects 4X1 group output for AUXSIG0 of CLB-XBAR<br>00 : Select .0 input for MUX6<br>01 : Select .1 input for MUX6<br>10 : Select .2 input for MUX6<br>11 : Select .3 input for MUX6<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX5 | R/W | 0h | Group Select Bits for MUX5: Selects 4X1 group output for AUXSIG0 of CLB-XBAR<br>00 : Select .0 input for MUX5<br>01 : Select .1 input for MUX5<br>10 : Select .2 input for MUX5<br>11 : Select .3 input for MUX5<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX4 | R/W | 0h | Group Select Bits for MUX4: Selects 4X1 group output for AUXSIG0 of CLB-XBAR<br>00 : Select .0 input for MUX4<br>01 : Select .1 input for MUX4<br>10 : Select .2 input for MUX4<br>11 : Select .3 input for MUX4<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-61. AUXSIG0MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 7-6 | MUX3 | R/W | 0h | Group Select Bits for MUX3: Selects 4X1 group output for AUXSIG0 of CLB-XBAR<br>00 : Select .0 input for MUX3<br>01 : Select .1 input for MUX3<br>10 : Select .2 input for MUX3<br>11 : Select .3 input for MUX3<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX2 | R/W | 0h | Group Select Bits for MUX2: Selects 4X1 group output for AUXSIG0 of CLB-XBAR<br>00 : Select .0 input for MUX2<br>01 : Select .1 input for MUX2<br>10 : Select .2 input for MUX2<br>11 : Select .3 input for MUX2<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 3-2 | MUX1 | R/W | 0h | Group Select Bits for MUX1: Selects 4X1 group output for AUXSIG0 of CLB-XBAR<br>00 : Select .0 input for MUX1<br>01 : Select .1 input for MUX1<br>10 : Select .2 input for MUX1<br>11 : Select .3 input for MUX1<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX0 | R/W | 0h | Group Select Bits for MUX0: Selects 4X1 group output for AUXSIG0 of CLB-XBAR<br>00 : Select .0 input for MUX0<br>01 : Select .1 input for MUX0<br>10 : Select .2 input for MUX0<br>11 : Select .3 input for MUX0<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.5.2 AUXSIG0MUX16TO31CFG Register (Offset = 2h) [Reset = 00000000h]

AUXSIG0MUX16TO31CFG is shown in Figure 9-55 and described in Table 9-62.

Return to the Summary Table.

CLB XBAR Mux Configuration for Output-0

**Figure 9-55. AUXSIG0MUX16TO31CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX31 | | MUX30 | | MUX29 | | MUX28 | | MUX27 | | MUX26 | | MUX25 | | MUX24 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX23 | | MUX22 | | MUX21 | | MUX20 | | MUX19 | | MUX18 | | MUX17 | | MUX16 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-62. AUXSIG0MUX16TO31CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX31 | R/W | 0h | Group Select Bits for MUX31: Selects 4X1 group output for AUXSIG0 of CLB-XBAR<br>00 : Select .0 input for MUX31<br>01 : Select .1 input for MUX31<br>10 : Select .2 input for MUX31<br>11 : Select .3 input for MUX31<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX30 | R/W | 0h | Group Select Bits for MUX30: Selects 4X1 group output for AUXSIG0 of CLB-XBAR<br>00 : Select .0 input for MUX30<br>01 : Select .1 input for MUX30<br>10 : Select .2 input for MUX30<br>11 : Select .3 input for MUX30<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX29 | R/W | 0h | Group Select Bits for MUX29: Selects 4X1 group output for AUXSIG0 of CLB-XBAR<br>00 : Select .0 input for MUX29<br>01 : Select .1 input for MUX29<br>10 : Select .2 input for MUX29<br>11 : Select .3 input for MUX29<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX28 | R/W | 0h | Group Select Bits for MUX28: Selects 4X1 group output for AUXSIG0 of CLB-XBAR<br>00 : Select .0 input for MUX28<br>01 : Select .1 input for MUX28<br>10 : Select .2 input for MUX28<br>11 : Select .3 input for MUX28<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX27 | R/W | 0h | Group Select Bits for MUX27: Selects 4X1 group output for AUXSIG0 of CLB-XBAR<br>00 : Select .0 input for MUX27<br>01 : Select .1 input for MUX27<br>10 : Select .2 input for MUX27<br>11 : Select .3 input for MUX27<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-62. AUXSIG0MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 21-20 | MUX26 | R/W | 0h | Group Select Bits for MUX26: Selects 4X1 group output for AUXSIG0 of CLB-XBAR<br>00 : Select .0 input for MUX26<br>01 : Select .1 input for MUX26<br>10 : Select .2 input for MUX26<br>11 : Select .3 input for MUX26<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 19-18 | MUX25 | R/W | 0h | Group Select Bits for MUX25: Selects 4X1 group output for AUXSIG0 of CLB-XBAR<br>00 : Select .0 input for MUX25<br>01 : Select .1 input for MUX25<br>10 : Select .2 input for MUX25<br>11 : Select .3 input for MUX25<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX24 | R/W | 0h | Group Select Bits for MUX24: Selects 4X1 group output for AUXSIG0 of CLB-XBAR<br>00 : Select .0 input for MUX24<br>01 : Select .1 input for MUX24<br>10 : Select .2 input for MUX24<br>11 : Select .3 input for MUX24<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX23 | R/W | 0h | Group Select Bits for MUX23: Selects 4X1 group output for AUXSIG0 of CLB-XBAR<br>00 : Select .0 input for MUX23<br>01 : Select .1 input for MUX23<br>10 : Select .2 input for MUX23<br>11 : Select .3 input for MUX23<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX22 | R/W | 0h | Group Select Bits for MUX22: Selects 4X1 group output for AUXSIG0 of CLB-XBAR<br>00 : Select .0 input for MUX22<br>01 : Select .1 input for MUX22<br>10 : Select .2 input for MUX22<br>11 : Select .3 input for MUX22<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX21 | R/W | 0h | Group Select Bits for MUX21: Selects 4X1 group output for AUXSIG0 of CLB-XBAR<br>00 : Select .0 input for MUX21<br>01 : Select .1 input for MUX21<br>10 : Select .2 input for MUX21<br>11 : Select .3 input for MUX21<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX20 | R/W | 0h | Group Select Bits for MUX20: Selects 4X1 group output for AUXSIG0 of CLB-XBAR<br>00 : Select .0 input for MUX20<br>01 : Select .1 input for MUX20<br>10 : Select .2 input for MUX20<br>11 : Select .3 input for MUX20<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-62. AUXSIG0MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 7-6 | MUX19 | R/W | 0h | Group Select Bits for MUX19: Selects 4X1 group output for AUXSIG0 of CLB-XBAR<br>00 : Select .0 input for MUX19<br>01 : Select .1 input for MUX19<br>10 : Select .2 input for MUX19<br>11 : Select .3 input for MUX19<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX18 | R/W | 0h | Group Select Bits for MUX18: Selects 4X1 group output for AUXSIG0 of CLB-XBAR<br>00 : Select .0 input for MUX18<br>01 : Select .1 input for MUX18<br>10 : Select .2 input for MUX18<br>11 : Select .3 input for MUX18<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 3-2 | MUX17 | R/W | 0h | Group Select Bits for MUX17: Selects 4X1 group output for AUXSIG0 of CLB-XBAR<br>00 : Select .0 input for MUX17<br>01 : Select .1 input for MUX17<br>10 : Select .2 input for MUX17<br>11 : Select .3 input for MUX17<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX16 | R/W | 0h | Group Select Bits for MUX16: Selects 4X1 group output for AUXSIG0 of CLB-XBAR<br>00 : Select .0 input for MUX16<br>01 : Select .1 input for MUX16<br>10 : Select .2 input for MUX16<br>11 : Select .3 input for MUX16<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.5.3 AUXSIG1MUX0TO15CFG Register (Offset = 4h) [Reset = 00000000h]

AUXSIG1MUX0TO15CFG is shown in Figure 9-56 and described in Table 9-63.

Return to the Summary Table.

CLB XBAR Mux Configuration for Output-1

**Figure 9-56. AUXSIG1MUX0TO15CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX15 | | MUX14 | | MUX13 | | MUX12 | | MUX11 | | MUX10 | | MUX9 | | MUX8 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | | MUX6 | | MUX5 | | MUX4 | | MUX3 | | MUX2 | | MUX1 | | MUX0 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-63. AUXSIG1MUX0TO15CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX15 | R/W | 0h | Group Select Bits for MUX15: Selects 4X1 group output for AUXSIG1 of CLB-XBAR<br>00 : Select .0 input for MUX15<br>01 : Select .1 input for MUX15<br>10 : Select .2 input for MUX15<br>11 : Select .3 input for MUX15<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX14 | R/W | 0h | Group Select Bits for MUX14: Selects 4X1 group output for AUXSIG1 of CLB-XBAR<br>00 : Select .0 input for MUX14<br>01 : Select .1 input for MUX14<br>10 : Select .2 input for MUX14<br>11 : Select .3 input for MUX14<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX13 | R/W | 0h | Group Select Bits for MUX13: Selects 4X1 group output for AUXSIG1 of CLB-XBAR<br>00 : Select .0 input for MUX13<br>01 : Select .1 input for MUX13<br>10 : Select .2 input for MUX13<br>11 : Select .3 input for MUX13<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX12 | R/W | 0h | Group Select Bits for MUX12: Selects 4X1 group output for AUXSIG1 of CLB-XBAR<br>00 : Select .0 input for MUX12<br>01 : Select .1 input for MUX12<br>10 : Select .2 input for MUX12<br>11 : Select .3 input for MUX12<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX11 | R/W | 0h | Group Select Bits for MUX11: Selects 4X1 group output for AUXSIG1 of CLB-XBAR<br>00 : Select .0 input for MUX11<br>01 : Select .1 input for MUX11<br>10 : Select .2 input for MUX11<br>11 : Select .3 input for MUX11<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-63. AUXSIG1MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 21-20 | MUX10 | R/W | 0h | Group Select Bits for MUX10: Selects 4X1 group output for AUXSIG1 of CLB-XBAR<br>00 : Select .0 input for MUX10<br>01 : Select .1 input for MUX10<br>10 : Select .2 input for MUX10<br>11 : Select .3 input for MUX10<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 19-18 | MUX9 | R/W | 0h | Group Select Bits for MUX9: Selects 4X1 group output for AUXSIG1 of CLB-XBAR<br>00 : Select .0 input for MUX9<br>01 : Select .1 input for MUX9<br>10 : Select .2 input for MUX9<br>11 : Select .3 input for MUX9<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX8 | R/W | 0h | Group Select Bits for MUX8: Selects 4X1 group output for AUXSIG1 of CLB-XBAR<br>00 : Select .0 input for MUX8<br>01 : Select .1 input for MUX8<br>10 : Select .2 input for MUX8<br>11 : Select .3 input for MUX8<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX7 | R/W | 0h | Group Select Bits for MUX7: Selects 4X1 group output for AUXSIG1 of CLB-XBAR<br>00 : Select .0 input for MUX7<br>01 : Select .1 input for MUX7<br>10 : Select .2 input for MUX7<br>11 : Select .3 input for MUX7<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX6 | R/W | 0h | Group Select Bits for MUX6: Selects 4X1 group output for AUXSIG1 of CLB-XBAR<br>00 : Select .0 input for MUX6<br>01 : Select .1 input for MUX6<br>10 : Select .2 input for MUX6<br>11 : Select .3 input for MUX6<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX5 | R/W | 0h | Group Select Bits for MUX5: Selects 4X1 group output for AUXSIG1 of CLB-XBAR<br>00 : Select .0 input for MUX5<br>01 : Select .1 input for MUX5<br>10 : Select .2 input for MUX5<br>11 : Select .3 input for MUX5<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX4 | R/W | 0h | Group Select Bits for MUX4: Selects 4X1 group output for AUXSIG1 of CLB-XBAR<br>00 : Select .0 input for MUX4<br>01 : Select .1 input for MUX4<br>10 : Select .2 input for MUX4<br>11 : Select .3 input for MUX4<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-63. AUXSIG1MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 7-6 | MUX3 | R/W | 0h | Group Select Bits for MUX3: Selects 4X1 group output for AUXSIG1 of CLB-XBAR<br>00 : Select .0 input for MUX3<br>01 : Select .1 input for MUX3<br>10 : Select .2 input for MUX3<br>11 : Select .3 input for MUX3<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX2 | R/W | 0h | Group Select Bits for MUX2: Selects 4X1 group output for AUXSIG1 of CLB-XBAR<br>00 : Select .0 input for MUX2<br>01 : Select .1 input for MUX2<br>10 : Select .2 input for MUX2<br>11 : Select .3 input for MUX2<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 3-2 | MUX1 | R/W | 0h | Group Select Bits for MUX1: Selects 4X1 group output for AUXSIG1 of CLB-XBAR<br>00 : Select .0 input for MUX1<br>01 : Select .1 input for MUX1<br>10 : Select .2 input for MUX1<br>11 : Select .3 input for MUX1<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX0 | R/W | 0h | Group Select Bits for MUX0: Selects 4X1 group output for AUXSIG1 of CLB-XBAR<br>00 : Select .0 input for MUX0<br>01 : Select .1 input for MUX0<br>10 : Select .2 input for MUX0<br>11 : Select .3 input for MUX0<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.5.4 AUXSIG1MUX16TO31CFG Register (Offset = 6h) [Reset = 00000000h]

AUXSIG1MUX16TO31CFG is shown in Figure 9-57 and described in Table 9-64.

Return to the Summary Table.

CLB XBAR Mux Configuration for Output-1

**Figure 9-57. AUXSIG1MUX16TO31CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX31 | | MUX30 | | MUX29 | | MUX28 | | MUX27 | | MUX26 | | MUX25 | | MUX24 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX23 | | MUX22 | | MUX21 | | MUX20 | | MUX19 | | MUX18 | | MUX17 | | MUX16 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-64. AUXSIG1MUX16TO31CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX31 | R/W | 0h | Group Select Bits for MUX31: Selects 4X1 group output for AUXSIG1 of CLB-XBAR<br>00 : Select .0 input for MUX31<br>01 : Select .1 input for MUX31<br>10 : Select .2 input for MUX31<br>11 : Select .3 input for MUX31<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX30 | R/W | 0h | Group Select Bits for MUX30: Selects 4X1 group output for AUXSIG1 of CLB-XBAR<br>00 : Select .0 input for MUX30<br>01 : Select .1 input for MUX30<br>10 : Select .2 input for MUX30<br>11 : Select .3 input for MUX30<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX29 | R/W | 0h | Group Select Bits for MUX29: Selects 4X1 group output for AUXSIG1 of CLB-XBAR<br>00 : Select .0 input for MUX29<br>01 : Select .1 input for MUX29<br>10 : Select .2 input for MUX29<br>11 : Select .3 input for MUX29<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX28 | R/W | 0h | Group Select Bits for MUX28: Selects 4X1 group output for AUXSIG1 of CLB-XBAR<br>00 : Select .0 input for MUX28<br>01 : Select .1 input for MUX28<br>10 : Select .2 input for MUX28<br>11 : Select .3 input for MUX28<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX27 | R/W | 0h | Group Select Bits for MUX27: Selects 4X1 group output for AUXSIG1 of CLB-XBAR<br>00 : Select .0 input for MUX27<br>01 : Select .1 input for MUX27<br>10 : Select .2 input for MUX27<br>11 : Select .3 input for MUX27<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-64. AUXSIG1MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 21-20 | MUX26 | R/W | 0h | Group Select Bits for MUX26: Selects 4X1 group output for AUXSIG1 of CLB-XBAR<br>00 : Select .0 input for MUX26<br>01 : Select .1 input for MUX26<br>10 : Select .2 input for MUX26<br>11 : Select .3 input for MUX26<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 19-18 | MUX25 | R/W | 0h | Group Select Bits for MUX25: Selects 4X1 group output for AUXSIG1 of CLB-XBAR<br>00 : Select .0 input for MUX25<br>01 : Select .1 input for MUX25<br>10 : Select .2 input for MUX25<br>11 : Select .3 input for MUX25<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX24 | R/W | 0h | Group Select Bits for MUX24: Selects 4X1 group output for AUXSIG1 of CLB-XBAR<br>00 : Select .0 input for MUX24<br>01 : Select .1 input for MUX24<br>10 : Select .2 input for MUX24<br>11 : Select .3 input for MUX24<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX23 | R/W | 0h | Group Select Bits for MUX23: Selects 4X1 group output for AUXSIG1 of CLB-XBAR<br>00 : Select .0 input for MUX23<br>01 : Select .1 input for MUX23<br>10 : Select .2 input for MUX23<br>11 : Select .3 input for MUX23<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX22 | R/W | 0h | Group Select Bits for MUX22: Selects 4X1 group output for AUXSIG1 of CLB-XBAR<br>00 : Select .0 input for MUX22<br>01 : Select .1 input for MUX22<br>10 : Select .2 input for MUX22<br>11 : Select .3 input for MUX22<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX21 | R/W | 0h | Group Select Bits for MUX21: Selects 4X1 group output for AUXSIG1 of CLB-XBAR<br>00 : Select .0 input for MUX21<br>01 : Select .1 input for MUX21<br>10 : Select .2 input for MUX21<br>11 : Select .3 input for MUX21<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX20 | R/W | 0h | Group Select Bits for MUX20: Selects 4X1 group output for AUXSIG1 of CLB-XBAR<br>00 : Select .0 input for MUX20<br>01 : Select .1 input for MUX20<br>10 : Select .2 input for MUX20<br>11 : Select .3 input for MUX20<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-64. AUXSIG1MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 7-6 | MUX19 | R/W | 0h | Group Select Bits for MUX19: Selects 4X1 group output for AUXSIG1 of CLB-XBAR<br>00 : Select .0 input for MUX19<br>01 : Select .1 input for MUX19<br>10 : Select .2 input for MUX19<br>11 : Select .3 input for MUX19<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX18 | R/W | 0h | Group Select Bits for MUX18: Selects 4X1 group output for AUXSIG1 of CLB-XBAR<br>00 : Select .0 input for MUX18<br>01 : Select .1 input for MUX18<br>10 : Select .2 input for MUX18<br>11 : Select .3 input for MUX18<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 3-2 | MUX17 | R/W | 0h | Group Select Bits for MUX17: Selects 4X1 group output for AUXSIG1 of CLB-XBAR<br>00 : Select .0 input for MUX17<br>01 : Select .1 input for MUX17<br>10 : Select .2 input for MUX17<br>11 : Select .3 input for MUX17<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX16 | R/W | 0h | Group Select Bits for MUX16: Selects 4X1 group output for AUXSIG1 of CLB-XBAR<br>00 : Select .0 input for MUX16<br>01 : Select .1 input for MUX16<br>10 : Select .2 input for MUX16<br>11 : Select .3 input for MUX16<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.5.5 AUXSIG2MUX0TO15CFG Register (Offset = 8h) [Reset = 00000000h]

AUXSIG2MUX0TO15CFG is shown in Figure 9-58 and described in Table 9-65.

Return to the Summary Table.

CLB XBAR Mux Configuration for Output-2

**Figure 9-58. AUXSIG2MUX0TO15CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX15 | | MUX14 | | MUX13 | | MUX12 | | MUX11 | | MUX10 | | MUX9 | | MUX8 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | | MUX6 | | MUX5 | | MUX4 | | MUX3 | | MUX2 | | MUX1 | | MUX0 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-65. AUXSIG2MUX0TO15CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX15 | R/W | 0h | Group Select Bits for MUX15: Selects 4X1 group output for AUXSIG2 of CLB-XBAR<br>00 : Select .0 input for MUX15<br>01 : Select .1 input for MUX15<br>10 : Select .2 input for MUX15<br>11 : Select .3 input for MUX15<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX14 | R/W | 0h | Group Select Bits for MUX14: Selects 4X1 group output for AUXSIG2 of CLB-XBAR<br>00 : Select .0 input for MUX14<br>01 : Select .1 input for MUX14<br>10 : Select .2 input for MUX14<br>11 : Select .3 input for MUX14<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX13 | R/W | 0h | Group Select Bits for MUX13: Selects 4X1 group output for AUXSIG2 of CLB-XBAR<br>00 : Select .0 input for MUX13<br>01 : Select .1 input for MUX13<br>10 : Select .2 input for MUX13<br>11 : Select .3 input for MUX13<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX12 | R/W | 0h | Group Select Bits for MUX12: Selects 4X1 group output for AUXSIG2 of CLB-XBAR<br>00 : Select .0 input for MUX12<br>01 : Select .1 input for MUX12<br>10 : Select .2 input for MUX12<br>11 : Select .3 input for MUX12<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX11 | R/W | 0h | Group Select Bits for MUX11: Selects 4X1 group output for AUXSIG2 of CLB-XBAR<br>00 : Select .0 input for MUX11<br>01 : Select .1 input for MUX11<br>10 : Select .2 input for MUX11<br>11 : Select .3 input for MUX11<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-65. AUXSIG2MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 21-20 | MUX10 | R/W | 0h | Group Select Bits for MUX10: Selects 4X1 group output for AUXSIG2 of CLB-XBAR<br>00 : Select .0 input for MUX10<br>01 : Select .1 input for MUX10<br>10 : Select .2 input for MUX10<br>11 : Select .3 input for MUX10<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 19-18 | MUX9 | R/W | 0h | Group Select Bits for MUX9: Selects 4X1 group output for AUXSIG2 of CLB-XBAR<br>00 : Select .0 input for MUX9<br>01 : Select .1 input for MUX9<br>10 : Select .2 input for MUX9<br>11 : Select .3 input for MUX9<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX8 | R/W | 0h | Group Select Bits for MUX8: Selects 4X1 group output for AUXSIG2 of CLB-XBAR<br>00 : Select .0 input for MUX8<br>01 : Select .1 input for MUX8<br>10 : Select .2 input for MUX8<br>11 : Select .3 input for MUX8<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX7 | R/W | 0h | Group Select Bits for MUX7: Selects 4X1 group output for AUXSIG2 of CLB-XBAR<br>00 : Select .0 input for MUX7<br>01 : Select .1 input for MUX7<br>10 : Select .2 input for MUX7<br>11 : Select .3 input for MUX7<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX6 | R/W | 0h | Group Select Bits for MUX6: Selects 4X1 group output for AUXSIG2 of CLB-XBAR<br>00 : Select .0 input for MUX6<br>01 : Select .1 input for MUX6<br>10 : Select .2 input for MUX6<br>11 : Select .3 input for MUX6<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX5 | R/W | 0h | Group Select Bits for MUX5: Selects 4X1 group output for AUXSIG2 of CLB-XBAR<br>00 : Select .0 input for MUX5<br>01 : Select .1 input for MUX5<br>10 : Select .2 input for MUX5<br>11 : Select .3 input for MUX5<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX4 | R/W | 0h | Group Select Bits for MUX4: Selects 4X1 group output for AUXSIG2 of CLB-XBAR<br>00 : Select .0 input for MUX4<br>01 : Select .1 input for MUX4<br>10 : Select .2 input for MUX4<br>11 : Select .3 input for MUX4<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-65. AUXSIG2MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 7-6 | MUX3 | R/W | 0h | Group Select Bits for MUX3: Selects 4X1 group output for AUXSIG2 of CLB-XBAR<br>00 : Select .0 input for MUX3<br>01 : Select .1 input for MUX3<br>10 : Select .2 input for MUX3<br>11 : Select .3 input for MUX3<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX2 | R/W | 0h | Group Select Bits for MUX2: Selects 4X1 group output for AUXSIG2 of CLB-XBAR<br>00 : Select .0 input for MUX2<br>01 : Select .1 input for MUX2<br>10 : Select .2 input for MUX2<br>11 : Select .3 input for MUX2<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 3-2 | MUX1 | R/W | 0h | Group Select Bits for MUX1: Selects 4X1 group output for AUXSIG2 of CLB-XBAR<br>00 : Select .0 input for MUX1<br>01 : Select .1 input for MUX1<br>10 : Select .2 input for MUX1<br>11 : Select .3 input for MUX1<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX0 | R/W | 0h | Group Select Bits for MUX0: Selects 4X1 group output for AUXSIG2 of CLB-XBAR<br>00 : Select .0 input for MUX0<br>01 : Select .1 input for MUX0<br>10 : Select .2 input for MUX0<br>11 : Select .3 input for MUX0<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.5.6 AUXSIG2MUX16TO31CFG Register (Offset = Ah) [Reset = 00000000h]

AUXSIG2MUX16TO31CFG is shown in Figure 9-59 and described in Table 9-66.

Return to the Summary Table.

CLB XBAR Mux Configuration for Output-2

**Figure 9-59. AUXSIG2MUX16TO31CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX31 | | MUX30 | | MUX29 | | MUX28 | | MUX27 | | MUX26 | | MUX25 | | MUX24 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX23 | | MUX22 | | MUX21 | | MUX20 | | MUX19 | | MUX18 | | MUX17 | | MUX16 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-66. AUXSIG2MUX16TO31CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX31 | R/W | 0h | Group Select Bits for MUX31: Selects 4X1 group output for AUXSIG2 of CLB-XBAR<br>00 : Select .0 input for MUX31<br>01 : Select .1 input for MUX31<br>10 : Select .2 input for MUX31<br>11 : Select .3 input for MUX31<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX30 | R/W | 0h | Group Select Bits for MUX30: Selects 4X1 group output for AUXSIG2 of CLB-XBAR<br>00 : Select .0 input for MUX30<br>01 : Select .1 input for MUX30<br>10 : Select .2 input for MUX30<br>11 : Select .3 input for MUX30<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX29 | R/W | 0h | Group Select Bits for MUX29: Selects 4X1 group output for AUXSIG2 of CLB-XBAR<br>00 : Select .0 input for MUX29<br>01 : Select .1 input for MUX29<br>10 : Select .2 input for MUX29<br>11 : Select .3 input for MUX29<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX28 | R/W | 0h | Group Select Bits for MUX28: Selects 4X1 group output for AUXSIG2 of CLB-XBAR<br>00 : Select .0 input for MUX28<br>01 : Select .1 input for MUX28<br>10 : Select .2 input for MUX28<br>11 : Select .3 input for MUX28<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX27 | R/W | 0h | Group Select Bits for MUX27: Selects 4X1 group output for AUXSIG2 of CLB-XBAR<br>00 : Select .0 input for MUX27<br>01 : Select .1 input for MUX27<br>10 : Select .2 input for MUX27<br>11 : Select .3 input for MUX27<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-66. AUXSIG2MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 21-20 | MUX26 | R/W | 0h | Group Select Bits for MUX26: Selects 4X1 group output for AUXSIG2 of CLB-XBAR<br>00 : Select .0 input for MUX26<br>01 : Select .1 input for MUX26<br>10 : Select .2 input for MUX26<br>11 : Select .3 input for MUX26<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 19-18 | MUX25 | R/W | 0h | Group Select Bits for MUX25: Selects 4X1 group output for AUXSIG2 of CLB-XBAR<br>00 : Select .0 input for MUX25<br>01 : Select .1 input for MUX25<br>10 : Select .2 input for MUX25<br>11 : Select .3 input for MUX25<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX24 | R/W | 0h | Group Select Bits for MUX24: Selects 4X1 group output for AUXSIG2 of CLB-XBAR<br>00 : Select .0 input for MUX24<br>01 : Select .1 input for MUX24<br>10 : Select .2 input for MUX24<br>11 : Select .3 input for MUX24<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX23 | R/W | 0h | Group Select Bits for MUX23: Selects 4X1 group output for AUXSIG2 of CLB-XBAR<br>00 : Select .0 input for MUX23<br>01 : Select .1 input for MUX23<br>10 : Select .2 input for MUX23<br>11 : Select .3 input for MUX23<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX22 | R/W | 0h | Group Select Bits for MUX22: Selects 4X1 group output for AUXSIG2 of CLB-XBAR<br>00 : Select .0 input for MUX22<br>01 : Select .1 input for MUX22<br>10 : Select .2 input for MUX22<br>11 : Select .3 input for MUX22<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX21 | R/W | 0h | Group Select Bits for MUX21: Selects 4X1 group output for AUXSIG2 of CLB-XBAR<br>00 : Select .0 input for MUX21<br>01 : Select .1 input for MUX21<br>10 : Select .2 input for MUX21<br>11 : Select .3 input for MUX21<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX20 | R/W | 0h | Group Select Bits for MUX20: Selects 4X1 group output for AUXSIG2 of CLB-XBAR<br>00 : Select .0 input for MUX20<br>01 : Select .1 input for MUX20<br>10 : Select .2 input for MUX20<br>11 : Select .3 input for MUX20<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-66. AUXSIG2MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 7-6 | MUX19 | R/W | 0h | Group Select Bits for MUX19: Selects 4X1 group output for AUXSIG2 of CLB-XBAR<br>00 : Select .0 input for MUX19<br>01 : Select .1 input for MUX19<br>10 : Select .2 input for MUX19<br>11 : Select .3 input for MUX19<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX18 | R/W | 0h | Group Select Bits for MUX18: Selects 4X1 group output for AUXSIG2 of CLB-XBAR<br>00 : Select .0 input for MUX18<br>01 : Select .1 input for MUX18<br>10 : Select .2 input for MUX18<br>11 : Select .3 input for MUX18<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 3-2 | MUX17 | R/W | 0h | Group Select Bits for MUX17: Selects 4X1 group output for AUXSIG2 of CLB-XBAR<br>00 : Select .0 input for MUX17<br>01 : Select .1 input for MUX17<br>10 : Select .2 input for MUX17<br>11 : Select .3 input for MUX17<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX16 | R/W | 0h | Group Select Bits for MUX16: Selects 4X1 group output for AUXSIG2 of CLB-XBAR<br>00 : Select .0 input for MUX16<br>01 : Select .1 input for MUX16<br>10 : Select .2 input for MUX16<br>11 : Select .3 input for MUX16<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.5.7 AUXSIG3MUX0TO15CFG Register (Offset = Ch) [Reset = 00000000h]

AUXSIG3MUX0TO15CFG is shown in Figure 9-60 and described in Table 9-67.

Return to the Summary Table.

CLB XBAR Mux Configuration for Output-3

**Figure 9-60. AUXSIG3MUX0TO15CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX15 | | MUX14 | | MUX13 | | MUX12 | | MUX11 | | MUX10 | | MUX9 | | MUX8 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | | MUX6 | | MUX5 | | MUX4 | | MUX3 | | MUX2 | | MUX1 | | MUX0 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-67. AUXSIG3MUX0TO15CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX15 | R/W | 0h | Group Select Bits for MUX15: Selects 4X1 group output for AUXSIG3 of CLB-XBAR<br>00 : Select .0 input for MUX15<br>01 : Select .1 input for MUX15<br>10 : Select .2 input for MUX15<br>11 : Select .3 input for MUX15<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX14 | R/W | 0h | Group Select Bits for MUX14: Selects 4X1 group output for AUXSIG3 of CLB-XBAR<br>00 : Select .0 input for MUX14<br>01 : Select .1 input for MUX14<br>10 : Select .2 input for MUX14<br>11 : Select .3 input for MUX14<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX13 | R/W | 0h | Group Select Bits for MUX13: Selects 4X1 group output for AUXSIG3 of CLB-XBAR<br>00 : Select .0 input for MUX13<br>01 : Select .1 input for MUX13<br>10 : Select .2 input for MUX13<br>11 : Select .3 input for MUX13<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX12 | R/W | 0h | Group Select Bits for MUX12: Selects 4X1 group output for AUXSIG3 of CLB-XBAR<br>00 : Select .0 input for MUX12<br>01 : Select .1 input for MUX12<br>10 : Select .2 input for MUX12<br>11 : Select .3 input for MUX12<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX11 | R/W | 0h | Group Select Bits for MUX11: Selects 4X1 group output for AUXSIG3 of CLB-XBAR<br>00 : Select .0 input for MUX11<br>01 : Select .1 input for MUX11<br>10 : Select .2 input for MUX11<br>11 : Select .3 input for MUX11<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-67. AUXSIG3MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 21-20 | MUX10 | R/W | 0h | Group Select Bits for MUX10: Selects 4X1 group output for AUXSIG3 of CLB-XBAR<br>00 : Select .0 input for MUX10<br>01 : Select .1 input for MUX10<br>10 : Select .2 input for MUX10<br>11 : Select .3 input for MUX10<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 19-18 | MUX9 | R/W | 0h | Group Select Bits for MUX9: Selects 4X1 group output for AUXSIG3 of CLB-XBAR<br>00 : Select .0 input for MUX9<br>01 : Select .1 input for MUX9<br>10 : Select .2 input for MUX9<br>11 : Select .3 input for MUX9<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX8 | R/W | 0h | Group Select Bits for MUX8: Selects 4X1 group output for AUXSIG3 of CLB-XBAR<br>00 : Select .0 input for MUX8<br>01 : Select .1 input for MUX8<br>10 : Select .2 input for MUX8<br>11 : Select .3 input for MUX8<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX7 | R/W | 0h | Group Select Bits for MUX7: Selects 4X1 group output for AUXSIG3 of CLB-XBAR<br>00 : Select .0 input for MUX7<br>01 : Select .1 input for MUX7<br>10 : Select .2 input for MUX7<br>11 : Select .3 input for MUX7<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX6 | R/W | 0h | Group Select Bits for MUX6: Selects 4X1 group output for AUXSIG3 of CLB-XBAR<br>00 : Select .0 input for MUX6<br>01 : Select .1 input for MUX6<br>10 : Select .2 input for MUX6<br>11 : Select .3 input for MUX6<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX5 | R/W | 0h | Group Select Bits for MUX5: Selects 4X1 group output for AUXSIG3 of CLB-XBAR<br>00 : Select .0 input for MUX5<br>01 : Select .1 input for MUX5<br>10 : Select .2 input for MUX5<br>11 : Select .3 input for MUX5<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX4 | R/W | 0h | Group Select Bits for MUX4: Selects 4X1 group output for AUXSIG3 of CLB-XBAR<br>00 : Select .0 input for MUX4<br>01 : Select .1 input for MUX4<br>10 : Select .2 input for MUX4<br>11 : Select .3 input for MUX4<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-67. AUXSIG3MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 7-6 | MUX3 | R/W | 0h | Group Select Bits for MUX3: Selects 4X1 group output for AUXSIG3 of CLB-XBAR<br>00 : Select .0 input for MUX3<br>01 : Select .1 input for MUX3<br>10 : Select .2 input for MUX3<br>11 : Select .3 input for MUX3<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX2 | R/W | 0h | Group Select Bits for MUX2: Selects 4X1 group output for AUXSIG3 of CLB-XBAR<br>00 : Select .0 input for MUX2<br>01 : Select .1 input for MUX2<br>10 : Select .2 input for MUX2<br>11 : Select .3 input for MUX2<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 3-2 | MUX1 | R/W | 0h | Group Select Bits for MUX1: Selects 4X1 group output for AUXSIG3 of CLB-XBAR<br>00 : Select .0 input for MUX1<br>01 : Select .1 input for MUX1<br>10 : Select .2 input for MUX1<br>11 : Select .3 input for MUX1<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX0 | R/W | 0h | Group Select Bits for MUX0: Selects 4X1 group output for AUXSIG3 of CLB-XBAR<br>00 : Select .0 input for MUX0<br>01 : Select .1 input for MUX0<br>10 : Select .2 input for MUX0<br>11 : Select .3 input for MUX0<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.5.8 AUXSIG3MUX16TO31CFG Register (Offset = Eh) [Reset = 00000000h]

AUXSIG3MUX16TO31CFG is shown in Figure 9-61 and described in Table 9-68.

Return to the Summary Table.

CLB XBAR Mux Configuration for Output-3

**Figure 9-61. AUXSIG3MUX16TO31CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX31 | | MUX30 | | MUX29 | | MUX28 | | MUX27 | | MUX26 | | MUX25 | | MUX24 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX23 | | MUX22 | | MUX21 | | MUX20 | | MUX19 | | MUX18 | | MUX17 | | MUX16 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-68. AUXSIG3MUX16TO31CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX31 | R/W | 0h | Group Select Bits for MUX31: Selects 4X1 group output for AUXSIG3 of CLB-XBAR<br>00 : Select .0 input for MUX31<br>01 : Select .1 input for MUX31<br>10 : Select .2 input for MUX31<br>11 : Select .3 input for MUX31<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX30 | R/W | 0h | Group Select Bits for MUX30: Selects 4X1 group output for AUXSIG3 of CLB-XBAR<br>00 : Select .0 input for MUX30<br>01 : Select .1 input for MUX30<br>10 : Select .2 input for MUX30<br>11 : Select .3 input for MUX30<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX29 | R/W | 0h | Group Select Bits for MUX29: Selects 4X1 group output for AUXSIG3 of CLB-XBAR<br>00 : Select .0 input for MUX29<br>01 : Select .1 input for MUX29<br>10 : Select .2 input for MUX29<br>11 : Select .3 input for MUX29<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX28 | R/W | 0h | Group Select Bits for MUX28: Selects 4X1 group output for AUXSIG3 of CLB-XBAR<br>00 : Select .0 input for MUX28<br>01 : Select .1 input for MUX28<br>10 : Select .2 input for MUX28<br>11 : Select .3 input for MUX28<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX27 | R/W | 0h | Group Select Bits for MUX27: Selects 4X1 group output for AUXSIG3 of CLB-XBAR<br>00 : Select .0 input for MUX27<br>01 : Select .1 input for MUX27<br>10 : Select .2 input for MUX27<br>11 : Select .3 input for MUX27<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-68. AUXSIG3MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 21-20 | MUX26 | R/W | 0h | Group Select Bits for MUX26: Selects 4X1 group output for AUXSIG3 of CLB-XBAR<br>00 : Select .0 input for MUX26<br>01 : Select .1 input for MUX26<br>10 : Select .2 input for MUX26<br>11 : Select .3 input for MUX26<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 19-18 | MUX25 | R/W | 0h | Group Select Bits for MUX25: Selects 4X1 group output for AUXSIG3 of CLB-XBAR<br>00 : Select .0 input for MUX25<br>01 : Select .1 input for MUX25<br>10 : Select .2 input for MUX25<br>11 : Select .3 input for MUX25<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX24 | R/W | 0h | Group Select Bits for MUX24: Selects 4X1 group output for AUXSIG3 of CLB-XBAR<br>00 : Select .0 input for MUX24<br>01 : Select .1 input for MUX24<br>10 : Select .2 input for MUX24<br>11 : Select .3 input for MUX24<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX23 | R/W | 0h | Group Select Bits for MUX23: Selects 4X1 group output for AUXSIG3 of CLB-XBAR<br>00 : Select .0 input for MUX23<br>01 : Select .1 input for MUX23<br>10 : Select .2 input for MUX23<br>11 : Select .3 input for MUX23<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX22 | R/W | 0h | Group Select Bits for MUX22: Selects 4X1 group output for AUXSIG3 of CLB-XBAR<br>00 : Select .0 input for MUX22<br>01 : Select .1 input for MUX22<br>10 : Select .2 input for MUX22<br>11 : Select .3 input for MUX22<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX21 | R/W | 0h | Group Select Bits for MUX21: Selects 4X1 group output for AUXSIG3 of CLB-XBAR<br>00 : Select .0 input for MUX21<br>01 : Select .1 input for MUX21<br>10 : Select .2 input for MUX21<br>11 : Select .3 input for MUX21<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX20 | R/W | 0h | Group Select Bits for MUX20: Selects 4X1 group output for AUXSIG3 of CLB-XBAR<br>00 : Select .0 input for MUX20<br>01 : Select .1 input for MUX20<br>10 : Select .2 input for MUX20<br>11 : Select .3 input for MUX20<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-68. AUXSIG3MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 7-6 | MUX19 | R/W | 0h | Group Select Bits for MUX19: Selects 4X1 group output for AUXSIG3 of CLB-XBAR<br>00 : Select .0 input for MUX19<br>01 : Select .1 input for MUX19<br>10 : Select .2 input for MUX19<br>11 : Select .3 input for MUX19<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX18 | R/W | 0h | Group Select Bits for MUX18: Selects 4X1 group output for AUXSIG3 of CLB-XBAR<br>00 : Select .0 input for MUX18<br>01 : Select .1 input for MUX18<br>10 : Select .2 input for MUX18<br>11 : Select .3 input for MUX18<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 3-2 | MUX17 | R/W | 0h | Group Select Bits for MUX17: Selects 4X1 group output for AUXSIG3 of CLB-XBAR<br>00 : Select .0 input for MUX17<br>01 : Select .1 input for MUX17<br>10 : Select .2 input for MUX17<br>11 : Select .3 input for MUX17<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX16 | R/W | 0h | Group Select Bits for MUX16: Selects 4X1 group output for AUXSIG3 of CLB-XBAR<br>00 : Select .0 input for MUX16<br>01 : Select .1 input for MUX16<br>10 : Select .2 input for MUX16<br>11 : Select .3 input for MUX16<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.5.9 AUXSIG4MUX0TO15CFG Register (Offset = 10h) [Reset = 00000000h]

AUXSIG4MUX0TO15CFG is shown in Figure 9-62 and described in Table 9-69.

Return to the Summary Table.

CLB XBAR Mux Configuration for Output-4

**Figure 9-62. AUXSIG4MUX0TO15CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX15 | | MUX14 | | MUX13 | | MUX12 | | MUX11 | | MUX10 | | MUX9 | | MUX8 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | | MUX6 | | MUX5 | | MUX4 | | MUX3 | | MUX2 | | MUX1 | | MUX0 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-69. AUXSIG4MUX0TO15CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX15 | R/W | 0h | Group Select Bits for MUX15: Selects 4X1 group output for AUXSIG4 of CLB-XBAR<br>00 : Select .0 input for MUX15<br>01 : Select .1 input for MUX15<br>10 : Select .2 input for MUX15<br>11 : Select .3 input for MUX15<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX14 | R/W | 0h | Group Select Bits for MUX14: Selects 4X1 group output for AUXSIG4 of CLB-XBAR<br>00 : Select .0 input for MUX14<br>01 : Select .1 input for MUX14<br>10 : Select .2 input for MUX14<br>11 : Select .3 input for MUX14<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX13 | R/W | 0h | Group Select Bits for MUX13: Selects 4X1 group output for AUXSIG4 of CLB-XBAR<br>00 : Select .0 input for MUX13<br>01 : Select .1 input for MUX13<br>10 : Select .2 input for MUX13<br>11 : Select .3 input for MUX13<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX12 | R/W | 0h | Group Select Bits for MUX12: Selects 4X1 group output for AUXSIG4 of CLB-XBAR<br>00 : Select .0 input for MUX12<br>01 : Select .1 input for MUX12<br>10 : Select .2 input for MUX12<br>11 : Select .3 input for MUX12<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX11 | R/W | 0h | Group Select Bits for MUX11: Selects 4X1 group output for AUXSIG4 of CLB-XBAR<br>00 : Select .0 input for MUX11<br>01 : Select .1 input for MUX11<br>10 : Select .2 input for MUX11<br>11 : Select .3 input for MUX11<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-69. AUXSIG4MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 21-20 | MUX10 | R/W | 0h | Group Select Bits for MUX10: Selects 4X1 group output for AUXSIG4 of CLB-XBAR<br>00 : Select .0 input for MUX10<br>01 : Select .1 input for MUX10<br>10 : Select .2 input for MUX10<br>11 : Select .3 input for MUX10<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 19-18 | MUX9 | R/W | 0h | Group Select Bits for MUX9: Selects 4X1 group output for AUXSIG4 of CLB-XBAR<br>00 : Select .0 input for MUX9<br>01 : Select .1 input for MUX9<br>10 : Select .2 input for MUX9<br>11 : Select .3 input for MUX9<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX8 | R/W | 0h | Group Select Bits for MUX8: Selects 4X1 group output for AUXSIG4 of CLB-XBAR<br>00 : Select .0 input for MUX8<br>01 : Select .1 input for MUX8<br>10 : Select .2 input for MUX8<br>11 : Select .3 input for MUX8<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX7 | R/W | 0h | Group Select Bits for MUX7: Selects 4X1 group output for AUXSIG4 of CLB-XBAR<br>00 : Select .0 input for MUX7<br>01 : Select .1 input for MUX7<br>10 : Select .2 input for MUX7<br>11 : Select .3 input for MUX7<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX6 | R/W | 0h | Group Select Bits for MUX6: Selects 4X1 group output for AUXSIG4 of CLB-XBAR<br>00 : Select .0 input for MUX6<br>01 : Select .1 input for MUX6<br>10 : Select .2 input for MUX6<br>11 : Select .3 input for MUX6<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX5 | R/W | 0h | Group Select Bits for MUX5: Selects 4X1 group output for AUXSIG4 of CLB-XBAR<br>00 : Select .0 input for MUX5<br>01 : Select .1 input for MUX5<br>10 : Select .2 input for MUX5<br>11 : Select .3 input for MUX5<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX4 | R/W | 0h | Group Select Bits for MUX4: Selects 4X1 group output for AUXSIG4 of CLB-XBAR<br>00 : Select .0 input for MUX4<br>01 : Select .1 input for MUX4<br>10 : Select .2 input for MUX4<br>11 : Select .3 input for MUX4<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-69. AUXSIG4MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 7-6 | MUX3 | R/W | 0h | Group Select Bits for MUX3: Selects 4X1 group output for AUXSIG4 of CLB-XBAR<br>00 : Select .0 input for MUX3<br>01 : Select .1 input for MUX3<br>10 : Select .2 input for MUX3<br>11 : Select .3 input for MUX3<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX2 | R/W | 0h | Group Select Bits for MUX2: Selects 4X1 group output for AUXSIG4 of CLB-XBAR<br>00 : Select .0 input for MUX2<br>01 : Select .1 input for MUX2<br>10 : Select .2 input for MUX2<br>11 : Select .3 input for MUX2<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 3-2 | MUX1 | R/W | 0h | Group Select Bits for MUX1: Selects 4X1 group output for AUXSIG4 of CLB-XBAR<br>00 : Select .0 input for MUX1<br>01 : Select .1 input for MUX1<br>10 : Select .2 input for MUX1<br>11 : Select .3 input for MUX1<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX0 | R/W | 0h | Group Select Bits for MUX0: Selects 4X1 group output for AUXSIG4 of CLB-XBAR<br>00 : Select .0 input for MUX0<br>01 : Select .1 input for MUX0<br>10 : Select .2 input for MUX0<br>11 : Select .3 input for MUX0<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.5.10 AUXSIG4MUX16TO31CFG Register (Offset = 12h) [Reset = 00000000h]

AUXSIG4MUX16TO31CFG is shown in Figure 9-63 and described in Table 9-70.

Return to the Summary Table.

CLB XBAR Mux Configuration for Output-4

**Figure 9-63. AUXSIG4MUX16TO31CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX31 | | MUX30 | | MUX29 | | MUX28 | | MUX27 | | MUX26 | | MUX25 | | MUX24 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX23 | | MUX22 | | MUX21 | | MUX20 | | MUX19 | | MUX18 | | MUX17 | | MUX16 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-70. AUXSIG4MUX16TO31CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX31 | R/W | 0h | Group Select Bits for MUX31: Selects 4X1 group output for AUXSIG4 of CLB-XBAR<br>00 : Select .0 input for MUX31<br>01 : Select .1 input for MUX31<br>10 : Select .2 input for MUX31<br>11 : Select .3 input for MUX31<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX30 | R/W | 0h | Group Select Bits for MUX30: Selects 4X1 group output for AUXSIG4 of CLB-XBAR<br>00 : Select .0 input for MUX30<br>01 : Select .1 input for MUX30<br>10 : Select .2 input for MUX30<br>11 : Select .3 input for MUX30<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX29 | R/W | 0h | Group Select Bits for MUX29: Selects 4X1 group output for AUXSIG4 of CLB-XBAR<br>00 : Select .0 input for MUX29<br>01 : Select .1 input for MUX29<br>10 : Select .2 input for MUX29<br>11 : Select .3 input for MUX29<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX28 | R/W | 0h | Group Select Bits for MUX28: Selects 4X1 group output for AUXSIG4 of CLB-XBAR<br>00 : Select .0 input for MUX28<br>01 : Select .1 input for MUX28<br>10 : Select .2 input for MUX28<br>11 : Select .3 input for MUX28<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX27 | R/W | 0h | Group Select Bits for MUX27: Selects 4X1 group output for AUXSIG4 of CLB-XBAR<br>00 : Select .0 input for MUX27<br>01 : Select .1 input for MUX27<br>10 : Select .2 input for MUX27<br>11 : Select .3 input for MUX27<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-70. AUXSIG4MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 21-20 | MUX26 | R/W | 0h | Group Select Bits for MUX26: Selects 4X1 group output for AUXSIG4 of CLB-XBAR<br>00 : Select .0 input for MUX26<br>01 : Select .1 input for MUX26<br>10 : Select .2 input for MUX26<br>11 : Select .3 input for MUX26<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 19-18 | MUX25 | R/W | 0h | Group Select Bits for MUX25: Selects 4X1 group output for AUXSIG4 of CLB-XBAR<br>00 : Select .0 input for MUX25<br>01 : Select .1 input for MUX25<br>10 : Select .2 input for MUX25<br>11 : Select .3 input for MUX25<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX24 | R/W | 0h | Group Select Bits for MUX24: Selects 4X1 group output for AUXSIG4 of CLB-XBAR<br>00 : Select .0 input for MUX24<br>01 : Select .1 input for MUX24<br>10 : Select .2 input for MUX24<br>11 : Select .3 input for MUX24<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX23 | R/W | 0h | Group Select Bits for MUX23: Selects 4X1 group output for AUXSIG4 of CLB-XBAR<br>00 : Select .0 input for MUX23<br>01 : Select .1 input for MUX23<br>10 : Select .2 input for MUX23<br>11 : Select .3 input for MUX23<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX22 | R/W | 0h | Group Select Bits for MUX22: Selects 4X1 group output for AUXSIG4 of CLB-XBAR<br>00 : Select .0 input for MUX22<br>01 : Select .1 input for MUX22<br>10 : Select .2 input for MUX22<br>11 : Select .3 input for MUX22<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX21 | R/W | 0h | Group Select Bits for MUX21: Selects 4X1 group output for AUXSIG4 of CLB-XBAR<br>00 : Select .0 input for MUX21<br>01 : Select .1 input for MUX21<br>10 : Select .2 input for MUX21<br>11 : Select .3 input for MUX21<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX20 | R/W | 0h | Group Select Bits for MUX20: Selects 4X1 group output for AUXSIG4 of CLB-XBAR<br>00 : Select .0 input for MUX20<br>01 : Select .1 input for MUX20<br>10 : Select .2 input for MUX20<br>11 : Select .3 input for MUX20<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-70. AUXSIG4MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 7-6 | MUX19 | R/W | 0h | Group Select Bits for MUX19: Selects 4X1 group output for AUXSIG4 of CLB-XBAR<br>00 : Select .0 input for MUX19<br>01 : Select .1 input for MUX19<br>10 : Select .2 input for MUX19<br>11 : Select .3 input for MUX19<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX18 | R/W | 0h | Group Select Bits for MUX18: Selects 4X1 group output for AUXSIG4 of CLB-XBAR<br>00 : Select .0 input for MUX18<br>01 : Select .1 input for MUX18<br>10 : Select .2 input for MUX18<br>11 : Select .3 input for MUX18<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 3-2 | MUX17 | R/W | 0h | Group Select Bits for MUX17: Selects 4X1 group output for AUXSIG4 of CLB-XBAR<br>00 : Select .0 input for MUX17<br>01 : Select .1 input for MUX17<br>10 : Select .2 input for MUX17<br>11 : Select .3 input for MUX17<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX16 | R/W | 0h | Group Select Bits for MUX16: Selects 4X1 group output for AUXSIG4 of CLB-XBAR<br>00 : Select .0 input for MUX16<br>01 : Select .1 input for MUX16<br>10 : Select .2 input for MUX16<br>11 : Select .3 input for MUX16<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.5.11 AUXSIG5MUX0TO15CFG Register (Offset = 14h) [Reset = 00000000h]

AUXSIG5MUX0TO15CFG is shown in Figure 9-64 and described in Table 9-71.

Return to the Summary Table.

CLB XBAR Mux Configuration for Output-5

**Figure 9-64. AUXSIG5MUX0TO15CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX15 | | MUX14 | | MUX13 | | MUX12 | | MUX11 | | MUX10 | | MUX9 | | MUX8 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | | MUX6 | | MUX5 | | MUX4 | | MUX3 | | MUX2 | | MUX1 | | MUX0 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-71. AUXSIG5MUX0TO15CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX15 | R/W | 0h | Group Select Bits for MUX15: Selects 4X1 group output for AUXSIG5 of CLB-XBAR<br>00 : Select .0 input for MUX15<br>01 : Select .1 input for MUX15<br>10 : Select .2 input for MUX15<br>11 : Select .3 input for MUX15<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX14 | R/W | 0h | Group Select Bits for MUX14: Selects 4X1 group output for AUXSIG5 of CLB-XBAR<br>00 : Select .0 input for MUX14<br>01 : Select .1 input for MUX14<br>10 : Select .2 input for MUX14<br>11 : Select .3 input for MUX14<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX13 | R/W | 0h | Group Select Bits for MUX13: Selects 4X1 group output for AUXSIG5 of CLB-XBAR<br>00 : Select .0 input for MUX13<br>01 : Select .1 input for MUX13<br>10 : Select .2 input for MUX13<br>11 : Select .3 input for MUX13<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX12 | R/W | 0h | Group Select Bits for MUX12: Selects 4X1 group output for AUXSIG5 of CLB-XBAR<br>00 : Select .0 input for MUX12<br>01 : Select .1 input for MUX12<br>10 : Select .2 input for MUX12<br>11 : Select .3 input for MUX12<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX11 | R/W | 0h | Group Select Bits for MUX11: Selects 4X1 group output for AUXSIG5 of CLB-XBAR<br>00 : Select .0 input for MUX11<br>01 : Select .1 input for MUX11<br>10 : Select .2 input for MUX11<br>11 : Select .3 input for MUX11<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-71. AUXSIG5MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 21-20 | MUX10 | R/W | 0h | Group Select Bits for MUX10: Selects 4X1 group output for AUXSIG5 of CLB-XBAR<br>00 : Select .0 input for MUX10<br>01 : Select .1 input for MUX10<br>10 : Select .2 input for MUX10<br>11 : Select .3 input for MUX10<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 19-18 | MUX9 | R/W | 0h | Group Select Bits for MUX9: Selects 4X1 group output for AUXSIG5 of CLB-XBAR<br>00 : Select .0 input for MUX9<br>01 : Select .1 input for MUX9<br>10 : Select .2 input for MUX9<br>11 : Select .3 input for MUX9<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX8 | R/W | 0h | Group Select Bits for MUX8: Selects 4X1 group output for AUXSIG5 of CLB-XBAR<br>00 : Select .0 input for MUX8<br>01 : Select .1 input for MUX8<br>10 : Select .2 input for MUX8<br>11 : Select .3 input for MUX8<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX7 | R/W | 0h | Group Select Bits for MUX7: Selects 4X1 group output for AUXSIG5 of CLB-XBAR<br>00 : Select .0 input for MUX7<br>01 : Select .1 input for MUX7<br>10 : Select .2 input for MUX7<br>11 : Select .3 input for MUX7<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX6 | R/W | 0h | Group Select Bits for MUX6: Selects 4X1 group output for AUXSIG5 of CLB-XBAR<br>00 : Select .0 input for MUX6<br>01 : Select .1 input for MUX6<br>10 : Select .2 input for MUX6<br>11 : Select .3 input for MUX6<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX5 | R/W | 0h | Group Select Bits for MUX5: Selects 4X1 group output for AUXSIG5 of CLB-XBAR<br>00 : Select .0 input for MUX5<br>01 : Select .1 input for MUX5<br>10 : Select .2 input for MUX5<br>11 : Select .3 input for MUX5<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX4 | R/W | 0h | Group Select Bits for MUX4: Selects 4X1 group output for AUXSIG5 of CLB-XBAR<br>00 : Select .0 input for MUX4<br>01 : Select .1 input for MUX4<br>10 : Select .2 input for MUX4<br>11 : Select .3 input for MUX4<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-71. AUXSIG5MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 7-6 | MUX3 | R/W | 0h | Group Select Bits for MUX3: Selects 4X1 group output for AUXSIG5 of CLB-XBAR<br>00 : Select .0 input for MUX3<br>01 : Select .1 input for MUX3<br>10 : Select .2 input for MUX3<br>11 : Select .3 input for MUX3<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX2 | R/W | 0h | Group Select Bits for MUX2: Selects 4X1 group output for AUXSIG5 of CLB-XBAR<br>00 : Select .0 input for MUX2<br>01 : Select .1 input for MUX2<br>10 : Select .2 input for MUX2<br>11 : Select .3 input for MUX2<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 3-2 | MUX1 | R/W | 0h | Group Select Bits for MUX1: Selects 4X1 group output for AUXSIG5 of CLB-XBAR<br>00 : Select .0 input for MUX1<br>01 : Select .1 input for MUX1<br>10 : Select .2 input for MUX1<br>11 : Select .3 input for MUX1<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX0 | R/W | 0h | Group Select Bits for MUX0: Selects 4X1 group output for AUXSIG5 of CLB-XBAR<br>00 : Select .0 input for MUX0<br>01 : Select .1 input for MUX0<br>10 : Select .2 input for MUX0<br>11 : Select .3 input for MUX0<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.5.12 AUXSIG5MUX16TO31CFG Register (Offset = 16h) [Reset = 00000000h]

AUXSIG5MUX16TO31CFG is shown in Figure 9-65 and described in Table 9-72.

Return to the Summary Table.

CLB XBAR Mux Configuration for Output-5

**Figure 9-65. AUXSIG5MUX16TO31CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX31 | | MUX30 | | MUX29 | | MUX28 | | MUX27 | | MUX26 | | MUX25 | | MUX24 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX23 | | MUX22 | | MUX21 | | MUX20 | | MUX19 | | MUX18 | | MUX17 | | MUX16 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-72. AUXSIG5MUX16TO31CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX31 | R/W | 0h | Group Select Bits for MUX31: Selects 4X1 group output for AUXSIG5 of CLB-XBAR<br>00 : Select .0 input for MUX31<br>01 : Select .1 input for MUX31<br>10 : Select .2 input for MUX31<br>11 : Select .3 input for MUX31<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX30 | R/W | 0h | Group Select Bits for MUX30: Selects 4X1 group output for AUXSIG5 of CLB-XBAR<br>00 : Select .0 input for MUX30<br>01 : Select .1 input for MUX30<br>10 : Select .2 input for MUX30<br>11 : Select .3 input for MUX30<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX29 | R/W | 0h | Group Select Bits for MUX29: Selects 4X1 group output for AUXSIG5 of CLB-XBAR<br>00 : Select .0 input for MUX29<br>01 : Select .1 input for MUX29<br>10 : Select .2 input for MUX29<br>11 : Select .3 input for MUX29<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX28 | R/W | 0h | Group Select Bits for MUX28: Selects 4X1 group output for AUXSIG5 of CLB-XBAR<br>00 : Select .0 input for MUX28<br>01 : Select .1 input for MUX28<br>10 : Select .2 input for MUX28<br>11 : Select .3 input for MUX28<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX27 | R/W | 0h | Group Select Bits for MUX27: Selects 4X1 group output for AUXSIG5 of CLB-XBAR<br>00 : Select .0 input for MUX27<br>01 : Select .1 input for MUX27<br>10 : Select .2 input for MUX27<br>11 : Select .3 input for MUX27<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-72. AUXSIG5MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 21-20 | MUX26 | R/W | 0h | Group Select Bits for MUX26: Selects 4X1 group output for AUXSIG5 of CLB-XBAR<br>00 : Select .0 input for MUX26<br>01 : Select .1 input for MUX26<br>10 : Select .2 input for MUX26<br>11 : Select .3 input for MUX26<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 19-18 | MUX25 | R/W | 0h | Group Select Bits for MUX25: Selects 4X1 group output for AUXSIG5 of CLB-XBAR<br>00 : Select .0 input for MUX25<br>01 : Select .1 input for MUX25<br>10 : Select .2 input for MUX25<br>11 : Select .3 input for MUX25<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX24 | R/W | 0h | Group Select Bits for MUX24: Selects 4X1 group output for AUXSIG5 of CLB-XBAR<br>00 : Select .0 input for MUX24<br>01 : Select .1 input for MUX24<br>10 : Select .2 input for MUX24<br>11 : Select .3 input for MUX24<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX23 | R/W | 0h | Group Select Bits for MUX23: Selects 4X1 group output for AUXSIG5 of CLB-XBAR<br>00 : Select .0 input for MUX23<br>01 : Select .1 input for MUX23<br>10 : Select .2 input for MUX23<br>11 : Select .3 input for MUX23<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX22 | R/W | 0h | Group Select Bits for MUX22: Selects 4X1 group output for AUXSIG5 of CLB-XBAR<br>00 : Select .0 input for MUX22<br>01 : Select .1 input for MUX22<br>10 : Select .2 input for MUX22<br>11 : Select .3 input for MUX22<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX21 | R/W | 0h | Group Select Bits for MUX21: Selects 4X1 group output for AUXSIG5 of CLB-XBAR<br>00 : Select .0 input for MUX21<br>01 : Select .1 input for MUX21<br>10 : Select .2 input for MUX21<br>11 : Select .3 input for MUX21<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX20 | R/W | 0h | Group Select Bits for MUX20: Selects 4X1 group output for AUXSIG5 of CLB-XBAR<br>00 : Select .0 input for MUX20<br>01 : Select .1 input for MUX20<br>10 : Select .2 input for MUX20<br>11 : Select .3 input for MUX20<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-72. AUXSIG5MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 7-6 | MUX19 | R/W | 0h | Group Select Bits for MUX19: Selects 4X1 group output for AUXSIG5 of CLB-XBAR<br>00 : Select .0 input for MUX19<br>01 : Select .1 input for MUX19<br>10 : Select .2 input for MUX19<br>11 : Select .3 input for MUX19<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX18 | R/W | 0h | Group Select Bits for MUX18: Selects 4X1 group output for AUXSIG5 of CLB-XBAR<br>00 : Select .0 input for MUX18<br>01 : Select .1 input for MUX18<br>10 : Select .2 input for MUX18<br>11 : Select .3 input for MUX18<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 3-2 | MUX17 | R/W | 0h | Group Select Bits for MUX17: Selects 4X1 group output for AUXSIG5 of CLB-XBAR<br>00 : Select .0 input for MUX17<br>01 : Select .1 input for MUX17<br>10 : Select .2 input for MUX17<br>11 : Select .3 input for MUX17<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX16 | R/W | 0h | Group Select Bits for MUX16: Selects 4X1 group output for AUXSIG5 of CLB-XBAR<br>00 : Select .0 input for MUX16<br>01 : Select .1 input for MUX16<br>10 : Select .2 input for MUX16<br>11 : Select .3 input for MUX16<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.5.13 AUXSIG6MUX0TO15CFG Register (Offset = 18h) [Reset = 00000000h]

AUXSIG6MUX0TO15CFG is shown in Figure 9-66 and described in Table 9-73.

Return to the Summary Table.

CLB XBAR Mux Configuration for Output-6

**Figure 9-66. AUXSIG6MUX0TO15CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX15 | | MUX14 | | MUX13 | | MUX12 | | MUX11 | | MUX10 | | MUX9 | | MUX8 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | | MUX6 | | MUX5 | | MUX4 | | MUX3 | | MUX2 | | MUX1 | | MUX0 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-73. AUXSIG6MUX0TO15CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX15 | R/W | 0h | Group Select Bits for MUX15: Selects 4X1 group output for AUXSIG6 of CLB-XBAR<br>00 : Select .0 input for MUX15<br>01 : Select .1 input for MUX15<br>10 : Select .2 input for MUX15<br>11 : Select .3 input for MUX15<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX14 | R/W | 0h | Group Select Bits for MUX14: Selects 4X1 group output for AUXSIG6 of CLB-XBAR<br>00 : Select .0 input for MUX14<br>01 : Select .1 input for MUX14<br>10 : Select .2 input for MUX14<br>11 : Select .3 input for MUX14<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX13 | R/W | 0h | Group Select Bits for MUX13: Selects 4X1 group output for AUXSIG6 of CLB-XBAR<br>00 : Select .0 input for MUX13<br>01 : Select .1 input for MUX13<br>10 : Select .2 input for MUX13<br>11 : Select .3 input for MUX13<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX12 | R/W | 0h | Group Select Bits for MUX12: Selects 4X1 group output for AUXSIG6 of CLB-XBAR<br>00 : Select .0 input for MUX12<br>01 : Select .1 input for MUX12<br>10 : Select .2 input for MUX12<br>11 : Select .3 input for MUX12<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX11 | R/W | 0h | Group Select Bits for MUX11: Selects 4X1 group output for AUXSIG6 of CLB-XBAR<br>00 : Select .0 input for MUX11<br>01 : Select .1 input for MUX11<br>10 : Select .2 input for MUX11<br>11 : Select .3 input for MUX11<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-73. AUXSIG6MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 21-20 | MUX10 | R/W | 0h | Group Select Bits for MUX10: Selects 4X1 group output for AUXSIG6 of CLB-XBAR<br>00 : Select .0 input for MUX10<br>01 : Select .1 input for MUX10<br>10 : Select .2 input for MUX10<br>11 : Select .3 input for MUX10<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 19-18 | MUX9 | R/W | 0h | Group Select Bits for MUX9: Selects 4X1 group output for AUXSIG6 of CLB-XBAR<br>00 : Select .0 input for MUX9<br>01 : Select .1 input for MUX9<br>10 : Select .2 input for MUX9<br>11 : Select .3 input for MUX9<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX8 | R/W | 0h | Group Select Bits for MUX8: Selects 4X1 group output for AUXSIG6 of CLB-XBAR<br>00 : Select .0 input for MUX8<br>01 : Select .1 input for MUX8<br>10 : Select .2 input for MUX8<br>11 : Select .3 input for MUX8<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX7 | R/W | 0h | Group Select Bits for MUX7: Selects 4X1 group output for AUXSIG6 of CLB-XBAR<br>00 : Select .0 input for MUX7<br>01 : Select .1 input for MUX7<br>10 : Select .2 input for MUX7<br>11 : Select .3 input for MUX7<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX6 | R/W | 0h | Group Select Bits for MUX6: Selects 4X1 group output for AUXSIG6 of CLB-XBAR<br>00 : Select .0 input for MUX6<br>01 : Select .1 input for MUX6<br>10 : Select .2 input for MUX6<br>11 : Select .3 input for MUX6<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX5 | R/W | 0h | Group Select Bits for MUX5: Selects 4X1 group output for AUXSIG6 of CLB-XBAR<br>00 : Select .0 input for MUX5<br>01 : Select .1 input for MUX5<br>10 : Select .2 input for MUX5<br>11 : Select .3 input for MUX5<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX4 | R/W | 0h | Group Select Bits for MUX4: Selects 4X1 group output for AUXSIG6 of CLB-XBAR<br>00 : Select .0 input for MUX4<br>01 : Select .1 input for MUX4<br>10 : Select .2 input for MUX4<br>11 : Select .3 input for MUX4<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-73. AUXSIG6MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 7-6 | MUX3 | R/W | 0h | Group Select Bits for MUX3: Selects 4X1 group output for AUXSIG6 of CLB-XBAR<br>00 : Select .0 input for MUX3<br>01 : Select .1 input for MUX3<br>10 : Select .2 input for MUX3<br>11 : Select .3 input for MUX3<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX2 | R/W | 0h | Group Select Bits for MUX2: Selects 4X1 group output for AUXSIG6 of CLB-XBAR<br>00 : Select .0 input for MUX2<br>01 : Select .1 input for MUX2<br>10 : Select .2 input for MUX2<br>11 : Select .3 input for MUX2<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 3-2 | MUX1 | R/W | 0h | Group Select Bits for MUX1: Selects 4X1 group output for AUXSIG6 of CLB-XBAR<br>00 : Select .0 input for MUX1<br>01 : Select .1 input for MUX1<br>10 : Select .2 input for MUX1<br>11 : Select .3 input for MUX1<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX0 | R/W | 0h | Group Select Bits for MUX0: Selects 4X1 group output for AUXSIG6 of CLB-XBAR<br>00 : Select .0 input for MUX0<br>01 : Select .1 input for MUX0<br>10 : Select .2 input for MUX0<br>11 : Select .3 input for MUX0<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.5.14 AUXSIG6MUX16TO31CFG Register (Offset = 1Ah) [Reset = 00000000h]

AUXSIG6MUX16TO31CFG is shown in Figure 9-67 and described in Table 9-74.

Return to the Summary Table.

CLB XBAR Mux Configuration for Output-6

**Figure 9-67. AUXSIG6MUX16TO31CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX31 | | MUX30 | | MUX29 | | MUX28 | | MUX27 | | MUX26 | | MUX25 | | MUX24 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX23 | | MUX22 | | MUX21 | | MUX20 | | MUX19 | | MUX18 | | MUX17 | | MUX16 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-74. AUXSIG6MUX16TO31CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX31 | R/W | 0h | Group Select Bits for MUX31: Selects 4X1 group output for AUXSIG6 of CLB-XBAR<br>00 : Select .0 input for MUX31<br>01 : Select .1 input for MUX31<br>10 : Select .2 input for MUX31<br>11 : Select .3 input for MUX31<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX30 | R/W | 0h | Group Select Bits for MUX30: Selects 4X1 group output for AUXSIG6 of CLB-XBAR<br>00 : Select .0 input for MUX30<br>01 : Select .1 input for MUX30<br>10 : Select .2 input for MUX30<br>11 : Select .3 input for MUX30<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX29 | R/W | 0h | Group Select Bits for MUX29: Selects 4X1 group output for AUXSIG6 of CLB-XBAR<br>00 : Select .0 input for MUX29<br>01 : Select .1 input for MUX29<br>10 : Select .2 input for MUX29<br>11 : Select .3 input for MUX29<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX28 | R/W | 0h | Group Select Bits for MUX28: Selects 4X1 group output for AUXSIG6 of CLB-XBAR<br>00 : Select .0 input for MUX28<br>01 : Select .1 input for MUX28<br>10 : Select .2 input for MUX28<br>11 : Select .3 input for MUX28<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX27 | R/W | 0h | Group Select Bits for MUX27: Selects 4X1 group output for AUXSIG6 of CLB-XBAR<br>00 : Select .0 input for MUX27<br>01 : Select .1 input for MUX27<br>10 : Select .2 input for MUX27<br>11 : Select .3 input for MUX27<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-74. AUXSIG6MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 21-20 | MUX26 | R/W | 0h | Group Select Bits for MUX26: Selects 4X1 group output for AUXSIG6 of CLB-XBAR<br>00 : Select .0 input for MUX26<br>01 : Select .1 input for MUX26<br>10 : Select .2 input for MUX26<br>11 : Select .3 input for MUX26<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 19-18 | MUX25 | R/W | 0h | Group Select Bits for MUX25: Selects 4X1 group output for AUXSIG6 of CLB-XBAR<br>00 : Select .0 input for MUX25<br>01 : Select .1 input for MUX25<br>10 : Select .2 input for MUX25<br>11 : Select .3 input for MUX25<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX24 | R/W | 0h | Group Select Bits for MUX24: Selects 4X1 group output for AUXSIG6 of CLB-XBAR<br>00 : Select .0 input for MUX24<br>01 : Select .1 input for MUX24<br>10 : Select .2 input for MUX24<br>11 : Select .3 input for MUX24<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX23 | R/W | 0h | Group Select Bits for MUX23: Selects 4X1 group output for AUXSIG6 of CLB-XBAR<br>00 : Select .0 input for MUX23<br>01 : Select .1 input for MUX23<br>10 : Select .2 input for MUX23<br>11 : Select .3 input for MUX23<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX22 | R/W | 0h | Group Select Bits for MUX22: Selects 4X1 group output for AUXSIG6 of CLB-XBAR<br>00 : Select .0 input for MUX22<br>01 : Select .1 input for MUX22<br>10 : Select .2 input for MUX22<br>11 : Select .3 input for MUX22<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX21 | R/W | 0h | Group Select Bits for MUX21: Selects 4X1 group output for AUXSIG6 of CLB-XBAR<br>00 : Select .0 input for MUX21<br>01 : Select .1 input for MUX21<br>10 : Select .2 input for MUX21<br>11 : Select .3 input for MUX21<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX20 | R/W | 0h | Group Select Bits for MUX20: Selects 4X1 group output for AUXSIG6 of CLB-XBAR<br>00 : Select .0 input for MUX20<br>01 : Select .1 input for MUX20<br>10 : Select .2 input for MUX20<br>11 : Select .3 input for MUX20<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-74. AUXSIG6MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 7-6 | MUX19 | R/W | 0h | Group Select Bits for MUX19: Selects 4X1 group output for AUXSIG6 of CLB-XBAR<br>00 : Select .0 input for MUX19<br>01 : Select .1 input for MUX19<br>10 : Select .2 input for MUX19<br>11 : Select .3 input for MUX19<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX18 | R/W | 0h | Group Select Bits for MUX18: Selects 4X1 group output for AUXSIG6 of CLB-XBAR<br>00 : Select .0 input for MUX18<br>01 : Select .1 input for MUX18<br>10 : Select .2 input for MUX18<br>11 : Select .3 input for MUX18<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 3-2 | MUX17 | R/W | 0h | Group Select Bits for MUX17: Selects 4X1 group output for AUXSIG6 of CLB-XBAR<br>00 : Select .0 input for MUX17<br>01 : Select .1 input for MUX17<br>10 : Select .2 input for MUX17<br>11 : Select .3 input for MUX17<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX16 | R/W | 0h | Group Select Bits for MUX16: Selects 4X1 group output for AUXSIG6 of CLB-XBAR<br>00 : Select .0 input for MUX16<br>01 : Select .1 input for MUX16<br>10 : Select .2 input for MUX16<br>11 : Select .3 input for MUX16<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.5.15 AUXSIG7MUX0TO15CFG Register (Offset = 1Ch) [Reset = 00000000h]

AUXSIG7MUX0TO15CFG is shown in Figure 9-68 and described in Table 9-75.

Return to the Summary Table.

CLB XBAR Mux Configuration for Output-7

**Figure 9-68. AUXSIG7MUX0TO15CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX15 | | MUX14 | | MUX13 | | MUX12 | | MUX11 | | MUX10 | | MUX9 | | MUX8 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | | MUX6 | | MUX5 | | MUX4 | | MUX3 | | MUX2 | | MUX1 | | MUX0 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-75. AUXSIG7MUX0TO15CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX15 | R/W | 0h | Group Select Bits for MUX15: Selects 4X1 group output for AUXSIG7 of CLB-XBAR<br>00 : Select .0 input for MUX15<br>01 : Select .1 input for MUX15<br>10 : Select .2 input for MUX15<br>11 : Select .3 input for MUX15<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX14 | R/W | 0h | Group Select Bits for MUX14: Selects 4X1 group output for AUXSIG7 of CLB-XBAR<br>00 : Select .0 input for MUX14<br>01 : Select .1 input for MUX14<br>10 : Select .2 input for MUX14<br>11 : Select .3 input for MUX14<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX13 | R/W | 0h | Group Select Bits for MUX13: Selects 4X1 group output for AUXSIG7 of CLB-XBAR<br>00 : Select .0 input for MUX13<br>01 : Select .1 input for MUX13<br>10 : Select .2 input for MUX13<br>11 : Select .3 input for MUX13<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX12 | R/W | 0h | Group Select Bits for MUX12: Selects 4X1 group output for AUXSIG7 of CLB-XBAR<br>00 : Select .0 input for MUX12<br>01 : Select .1 input for MUX12<br>10 : Select .2 input for MUX12<br>11 : Select .3 input for MUX12<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX11 | R/W | 0h | Group Select Bits for MUX11: Selects 4X1 group output for AUXSIG7 of CLB-XBAR<br>00 : Select .0 input for MUX11<br>01 : Select .1 input for MUX11<br>10 : Select .2 input for MUX11<br>11 : Select .3 input for MUX11<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-75. AUXSIG7MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 21-20 | MUX10 | R/W | 0h | Group Select Bits for MUX10: Selects 4X1 group output for AUXSIG7 of CLB-XBAR<br>00 : Select .0 input for MUX10<br>01 : Select .1 input for MUX10<br>10 : Select .2 input for MUX10<br>11 : Select .3 input for MUX10<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 19-18 | MUX9 | R/W | 0h | Group Select Bits for MUX9: Selects 4X1 group output for AUXSIG7 of CLB-XBAR<br>00 : Select .0 input for MUX9<br>01 : Select .1 input for MUX9<br>10 : Select .2 input for MUX9<br>11 : Select .3 input for MUX9<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX8 | R/W | 0h | Group Select Bits for MUX8: Selects 4X1 group output for AUXSIG7 of CLB-XBAR<br>00 : Select .0 input for MUX8<br>01 : Select .1 input for MUX8<br>10 : Select .2 input for MUX8<br>11 : Select .3 input for MUX8<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX7 | R/W | 0h | Group Select Bits for MUX7: Selects 4X1 group output for AUXSIG7 of CLB-XBAR<br>00 : Select .0 input for MUX7<br>01 : Select .1 input for MUX7<br>10 : Select .2 input for MUX7<br>11 : Select .3 input for MUX7<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX6 | R/W | 0h | Group Select Bits for MUX6: Selects 4X1 group output for AUXSIG7 of CLB-XBAR<br>00 : Select .0 input for MUX6<br>01 : Select .1 input for MUX6<br>10 : Select .2 input for MUX6<br>11 : Select .3 input for MUX6<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX5 | R/W | 0h | Group Select Bits for MUX5: Selects 4X1 group output for AUXSIG7 of CLB-XBAR<br>00 : Select .0 input for MUX5<br>01 : Select .1 input for MUX5<br>10 : Select .2 input for MUX5<br>11 : Select .3 input for MUX5<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX4 | R/W | 0h | Group Select Bits for MUX4: Selects 4X1 group output for AUXSIG7 of CLB-XBAR<br>00 : Select .0 input for MUX4<br>01 : Select .1 input for MUX4<br>10 : Select .2 input for MUX4<br>11 : Select .3 input for MUX4<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-75. AUXSIG7MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 7-6 | MUX3 | R/W | 0h | Group Select Bits for MUX3: Selects 4X1 group output for AUXSIG7 of CLB-XBAR<br>00 : Select .0 input for MUX3<br>01 : Select .1 input for MUX3<br>10 : Select .2 input for MUX3<br>11 : Select .3 input for MUX3<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX2 | R/W | 0h | Group Select Bits for MUX2: Selects 4X1 group output for AUXSIG7 of CLB-XBAR<br>00 : Select .0 input for MUX2<br>01 : Select .1 input for MUX2<br>10 : Select .2 input for MUX2<br>11 : Select .3 input for MUX2<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 3-2 | MUX1 | R/W | 0h | Group Select Bits for MUX1: Selects 4X1 group output for AUXSIG7 of CLB-XBAR<br>00 : Select .0 input for MUX1<br>01 : Select .1 input for MUX1<br>10 : Select .2 input for MUX1<br>11 : Select .3 input for MUX1<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX0 | R/W | 0h | Group Select Bits for MUX0: Selects 4X1 group output for AUXSIG7 of CLB-XBAR<br>00 : Select .0 input for MUX0<br>01 : Select .1 input for MUX0<br>10 : Select .2 input for MUX0<br>11 : Select .3 input for MUX0<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.5.16 AUXSIG7MUX16TO31CFG Register (Offset = 1Eh) [Reset = 00000000h]

AUXSIG7MUX16TO31CFG is shown in Figure 9-69 and described in Table 9-76.

Return to the Summary Table.

CLB XBAR Mux Configuration for Output-7

**Figure 9-69. AUXSIG7MUX16TO31CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX31 | | MUX30 | | MUX29 | | MUX28 | | MUX27 | | MUX26 | | MUX25 | | MUX24 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX23 | | MUX22 | | MUX21 | | MUX20 | | MUX19 | | MUX18 | | MUX17 | | MUX16 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-76. AUXSIG7MUX16TO31CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX31 | R/W | 0h | Group Select Bits for MUX31: Selects 4X1 group output for AUXSIG7 of CLB-XBAR<br>00 : Select .0 input for MUX31<br>01 : Select .1 input for MUX31<br>10 : Select .2 input for MUX31<br>11 : Select .3 input for MUX31<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX30 | R/W | 0h | Group Select Bits for MUX30: Selects 4X1 group output for AUXSIG7 of CLB-XBAR<br>00 : Select .0 input for MUX30<br>01 : Select .1 input for MUX30<br>10 : Select .2 input for MUX30<br>11 : Select .3 input for MUX30<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX29 | R/W | 0h | Group Select Bits for MUX29: Selects 4X1 group output for AUXSIG7 of CLB-XBAR<br>00 : Select .0 input for MUX29<br>01 : Select .1 input for MUX29<br>10 : Select .2 input for MUX29<br>11 : Select .3 input for MUX29<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX28 | R/W | 0h | Group Select Bits for MUX28: Selects 4X1 group output for AUXSIG7 of CLB-XBAR<br>00 : Select .0 input for MUX28<br>01 : Select .1 input for MUX28<br>10 : Select .2 input for MUX28<br>11 : Select .3 input for MUX28<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX27 | R/W | 0h | Group Select Bits for MUX27: Selects 4X1 group output for AUXSIG7 of CLB-XBAR<br>00 : Select .0 input for MUX27<br>01 : Select .1 input for MUX27<br>10 : Select .2 input for MUX27<br>11 : Select .3 input for MUX27<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-76. AUXSIG7MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 21-20 | MUX26 | R/W | 0h | Group Select Bits for MUX26: Selects 4X1 group output for AUXSIG7 of CLB-XBAR<br>00 : Select .0 input for MUX26<br>01 : Select .1 input for MUX26<br>10 : Select .2 input for MUX26<br>11 : Select .3 input for MUX26<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 19-18 | MUX25 | R/W | 0h | Group Select Bits for MUX25: Selects 4X1 group output for AUXSIG7 of CLB-XBAR<br>00 : Select .0 input for MUX25<br>01 : Select .1 input for MUX25<br>10 : Select .2 input for MUX25<br>11 : Select .3 input for MUX25<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX24 | R/W | 0h | Group Select Bits for MUX24: Selects 4X1 group output for AUXSIG7 of CLB-XBAR<br>00 : Select .0 input for MUX24<br>01 : Select .1 input for MUX24<br>10 : Select .2 input for MUX24<br>11 : Select .3 input for MUX24<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX23 | R/W | 0h | Group Select Bits for MUX23: Selects 4X1 group output for AUXSIG7 of CLB-XBAR<br>00 : Select .0 input for MUX23<br>01 : Select .1 input for MUX23<br>10 : Select .2 input for MUX23<br>11 : Select .3 input for MUX23<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX22 | R/W | 0h | Group Select Bits for MUX22: Selects 4X1 group output for AUXSIG7 of CLB-XBAR<br>00 : Select .0 input for MUX22<br>01 : Select .1 input for MUX22<br>10 : Select .2 input for MUX22<br>11 : Select .3 input for MUX22<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX21 | R/W | 0h | Group Select Bits for MUX21: Selects 4X1 group output for AUXSIG7 of CLB-XBAR<br>00 : Select .0 input for MUX21<br>01 : Select .1 input for MUX21<br>10 : Select .2 input for MUX21<br>11 : Select .3 input for MUX21<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX20 | R/W | 0h | Group Select Bits for MUX20: Selects 4X1 group output for AUXSIG7 of CLB-XBAR<br>00 : Select .0 input for MUX20<br>01 : Select .1 input for MUX20<br>10 : Select .2 input for MUX20<br>11 : Select .3 input for MUX20<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-76. AUXSIG7MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 7-6 | MUX19 | R/W | 0h | Group Select Bits for MUX19: Selects 4X1 group output for AUXSIG7 of CLB-XBAR<br>00 : Select .0 input for MUX19<br>01 : Select .1 input for MUX19<br>10 : Select .2 input for MUX19<br>11 : Select .3 input for MUX19<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX18 | R/W | 0h | Group Select Bits for MUX18: Selects 4X1 group output for AUXSIG7 of CLB-XBAR<br>00 : Select .0 input for MUX18<br>01 : Select .1 input for MUX18<br>10 : Select .2 input for MUX18<br>11 : Select .3 input for MUX18<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 3-2 | MUX17 | R/W | 0h | Group Select Bits for MUX17: Selects 4X1 group output for AUXSIG7 of CLB-XBAR<br>00 : Select .0 input for MUX17<br>01 : Select .1 input for MUX17<br>10 : Select .2 input for MUX17<br>11 : Select .3 input for MUX17<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX16 | R/W | 0h | Group Select Bits for MUX16: Selects 4X1 group output for AUXSIG7 of CLB-XBAR<br>00 : Select .0 input for MUX16<br>01 : Select .1 input for MUX16<br>10 : Select .2 input for MUX16<br>11 : Select .3 input for MUX16<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.5.17 AUXSIG0MUXENABLE Register (Offset = 20h) [Reset = 00000000h]

AUXSIG0MUXENABLE is shown in Figure 9-70 and described in Table 9-77.

Return to the Summary Table.

CLB XBAR Mux Enable Register for Output-0

**Figure 9-70. AUXSIG0MUXENABLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MUX31 | MUX30 | MUX29 | MUX28 | MUX27 | MUX26 | MUX25 | MUX24 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MUX23 | MUX22 | MUX21 | MUX20 | MUX19 | MUX18 | MUX17 | MUX16 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MUX15 | MUX14 | MUX13 | MUX12 | MUX11 | MUX10 | MUX9 | MUX8 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | MUX6 | MUX5 | MUX4 | MUX3 | MUX2 | MUX1 | MUX0 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 9-77. AUXSIG0MUXENABLE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | MUX31 | R/W | 0h | Selects the output of MUX31 to drive AUXSIG0 of CLB-XBAR<br>0: Respective output of MUX31 is disabled to drive the AUXSIG0 of CLB-XBAR<br>1: Respective output of MUX31 is enabled to drive the AUXSIG0 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 30 | MUX30 | R/W | 0h | Selects the output of MUX30 to drive AUXSIG0 of CLB-XBAR<br>0: Respective output of MUX30 is disabled to drive the AUXSIG0 of CLB-XBAR<br>1: Respective output of MUX30 is enabled to drive the AUXSIG0 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29 | MUX29 | R/W | 0h | Selects the output of MUX29 to drive AUXSIG0 of CLB-XBAR<br>0: Respective output of MUX29 is disabled to drive the AUXSIG0 of CLB-XBAR<br>1: Respective output of MUX29 is enabled to drive the AUXSIG0 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 28 | MUX28 | R/W | 0h | Selects the output of MUX28 to drive AUXSIG0 of CLB-XBAR<br>0: Respective output of MUX28 is disabled to drive the AUXSIG0 of CLB-XBAR<br>1: Respective output of MUX28 is enabled to drive the AUXSIG0 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-77. AUXSIG0MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 27 | MUX27 | R/W | 0h | Selects the output of MUX27 to drive AUXSIG0 of CLB-XBAR<br>0: Respective output of MUX27 is disabled to drive the AUXSIG0 of CLB-XBAR<br>1: Respective output of MUX27 is enabled to drive the AUXSIG0 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 26 | MUX26 | R/W | 0h | Selects the output of MUX26 to drive AUXSIG0 of CLB-XBAR<br>0: Respective output of MUX26 is disabled to drive the AUXSIG0 of CLB-XBAR<br>1: Respective output of MUX26 is enabled to drive the AUXSIG0 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25 | MUX25 | R/W | 0h | Selects the output of MUX25 to drive AUXSIG0 of CLB-XBAR<br>0: Respective output of MUX25 is disabled to drive the AUXSIG0 of CLB-XBAR<br>1: Respective output of MUX25 is enabled to drive the AUXSIG0 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 24 | MUX24 | R/W | 0h | Selects the output of MUX24 to drive AUXSIG0 of CLB-XBAR<br>0: Respective output of MUX24 is disabled to drive the AUXSIG0 of CLB-XBAR<br>1: Respective output of MUX24 is enabled to drive the AUXSIG0 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23 | MUX23 | R/W | 0h | Selects the output of MUX23 to drive AUXSIG0 of CLB-XBAR<br>0: Respective output of MUX23 is disabled to drive the AUXSIG0 of CLB-XBAR<br>1: Respective output of MUX23 is enabled to drive the AUXSIG0 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 22 | MUX22 | R/W | 0h | Selects the output of MUX22 to drive AUXSIG0 of CLB-XBAR<br>0: Respective output of MUX22 is disabled to drive the AUXSIG0 of CLB-XBAR<br>1: Respective output of MUX22 is enabled to drive the AUXSIG0 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21 | MUX21 | R/W | 0h | Selects the output of MUX21 to drive AUXSIG0 of CLB-XBAR<br>0: Respective output of MUX21 is disabled to drive the AUXSIG0 of CLB-XBAR<br>1: Respective output of MUX21 is enabled to drive the AUXSIG0 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 20 | MUX20 | R/W | 0h | Selects the output of MUX20 to drive AUXSIG0 of CLB-XBAR<br>0: Respective output of MUX20 is disabled to drive the AUXSIG0 of CLB-XBAR<br>1: Respective output of MUX20 is enabled to drive the AUXSIG0 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-77. AUXSIG0MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19 | MUX19 | R/W | 0h | Selects the output of MUX19 to drive AUXSIG0 of CLB-XBAR<br>0: Respective output of MUX19 is disabled to drive the AUXSIG0 of CLB-XBAR<br>1: Respective output of MUX19 is enabled to drive the AUXSIG0 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 18 | MUX18 | R/W | 0h | Selects the output of MUX18 to drive AUXSIG0 of CLB-XBAR<br>0: Respective output of MUX18 is disabled to drive the AUXSIG0 of CLB-XBAR<br>1: Respective output of MUX18 is enabled to drive the AUXSIG0 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17 | MUX17 | R/W | 0h | Selects the output of MUX17 to drive AUXSIG0 of CLB-XBAR<br>0: Respective output of MUX17 is disabled to drive the AUXSIG0 of CLB-XBAR<br>1: Respective output of MUX17 is enabled to drive the AUXSIG0 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 16 | MUX16 | R/W | 0h | Selects the output of MUX16 to drive AUXSIG0 of CLB-XBAR<br>0: Respective output of MUX16 is disabled to drive the AUXSIG0 of CLB-XBAR<br>1: Respective output of MUX16 is enabled to drive the AUXSIG0 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15 | MUX15 | R/W | 0h | Selects the output of MUX15 to drive AUXSIG0 of CLB-XBAR<br>0: Respective output of MUX15 is disabled to drive the AUXSIG0 of CLB-XBAR<br>1: Respective output of MUX15 is enabled to drive the AUXSIG0 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 14 | MUX14 | R/W | 0h | Selects the output of MUX14 to drive AUXSIG0 of CLB-XBAR<br>0: Respective output of MUX14 is disabled to drive the AUXSIG0 of CLB-XBAR<br>1: Respective output of MUX14 is enabled to drive the AUXSIG0 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13 | MUX13 | R/W | 0h | Selects the output of MUX13 to drive AUXSIG0 of CLB-XBAR<br>0: Respective output of MUX13 is disabled to drive the AUXSIG0 of CLB-XBAR<br>1: Respective output of MUX13 is enabled to drive the AUXSIG0 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 12 | MUX12 | R/W | 0h | Selects the output of MUX12 to drive AUXSIG0 of CLB-XBAR<br>0: Respective output of MUX12 is disabled to drive the AUXSIG0 of CLB-XBAR<br>1: Respective output of MUX12 is enabled to drive the AUXSIG0 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-77. AUXSIG0MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 11 | MUX11 | R/W | 0h | Selects the output of MUX11 to drive AUXSIG0 of CLB-XBAR<br>0: Respective output of MUX11 is disabled to drive the AUXSIG0 of CLB-XBAR<br>1: Respective output of MUX11 is enabled to drive the AUXSIG0 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 10 | MUX10 | R/W | 0h | Selects the output of MUX10 to drive AUXSIG0 of CLB-XBAR<br>0: Respective output of MUX10 is disabled to drive the AUXSIG0 of CLB-XBAR<br>1: Respective output of MUX10 is enabled to drive the AUXSIG0 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9 | MUX9 | R/W | 0h | Selects the output of MUX9 to drive AUXSIG0 of CLB-XBAR<br>0: Respective output of MUX9 is disabled to drive the AUXSIG0 of CLB-XBAR<br>1: Respective output of MUX9 is enabled to drive the AUXSIG0 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 8 | MUX8 | R/W | 0h | Selects the output of MUX8 to drive AUXSIG0 of CLB-XBAR<br>0: Respective output of MUX8 is disabled to drive the AUXSIG0 of CLB-XBAR<br>1: Respective output of MUX8 is enabled to drive the AUXSIG0 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7 | MUX7 | R/W | 0h | Selects the output of MUX7 to drive AUXSIG0 of CLB-XBAR<br>0: Respective output of MUX7 is disabled to drive the AUXSIG0 of CLB-XBAR<br>1: Respective output of MUX7 is enabled to drive the AUXSIG0 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 6 | MUX6 | R/W | 0h | Selects the output of MUX6 to drive AUXSIG0 of CLB-XBAR<br>0: Respective output of MUX6 is disabled to drive the AUXSIG0 of CLB-XBAR<br>1: Respective output of MUX6 is enabled to drive the AUXSIG0 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5 | MUX5 | R/W | 0h | Selects the output of MUX5 to drive AUXSIG0 of CLB-XBAR<br>0: Respective output of MUX5 is disabled to drive the AUXSIG0 of CLB-XBAR<br>1: Respective output of MUX5 is enabled to drive the AUXSIG0 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 4 | MUX4 | R/W | 0h | Selects the output of MUX4 to drive AUXSIG0 of CLB-XBAR<br>0: Respective output of MUX4 is disabled to drive the AUXSIG0 of CLB-XBAR<br>1: Respective output of MUX4 is enabled to drive the AUXSIG0 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-77. AUXSIG0MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3 | MUX3 | R/W | 0h | Selects the output of MUX3 to drive AUXSIG0 of CLB-XBAR<br>0: Respective output of MUX3 is disabled to drive the AUXSIG0 of CLB-XBAR<br>1: Respective output of MUX3 is enabled to drive the AUXSIG0 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 2 | MUX2 | R/W | 0h | Selects the output of MUX2 to drive AUXSIG0 of CLB-XBAR<br>0: Respective output of MUX2 is disabled to drive the AUXSIG0 of CLB-XBAR<br>1: Respective output of MUX2 is enabled to drive the AUXSIG0 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1 | MUX1 | R/W | 0h | Selects the output of MUX1 to drive AUXSIG0 of CLB-XBAR<br>0: Respective output of MUX1 is disabled to drive the AUXSIG0 of CLB-XBAR<br>1: Respective output of MUX1 is enabled to drive the AUXSIG0 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 0 | MUX0 | R/W | 0h | Selects the output of mux0 to drive AUXSIG0 of CLB-XBAR<br>0: Respective output of MUX0 is disabled to drive the AUXSIG0 of CLB-XBAR<br>1: Respective output of MUX0 is enabled to drive the AUXSIG0 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.5.18 AUXSIG1MUXENABLE Register (Offset = 22h) [Reset = 00000000h]

AUXSIG1MUXENABLE is shown in Figure 9-71 and described in Table 9-78.

Return to the Summary Table.

CLB XBAR Mux Enable Register for Output-1

**Figure 9-71. AUXSIG1MUXENABLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MUX31 | MUX30 | MUX29 | MUX28 | MUX27 | MUX26 | MUX25 | MUX24 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MUX23 | MUX22 | MUX21 | MUX20 | MUX19 | MUX18 | MUX17 | MUX16 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MUX15 | MUX14 | MUX13 | MUX12 | MUX11 | MUX10 | MUX9 | MUX8 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | MUX6 | MUX5 | MUX4 | MUX3 | MUX2 | MUX1 | MUX0 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 9-78. AUXSIG1MUXENABLE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | MUX31 | R/W | 0h | Selects the output of MUX31 to drive AUXSIG1 of CLB-XBAR<br>0: Respective output of MUX31 is disabled to drive the AUXSIG1 of CLB-XBAR<br>1: Respective output of MUX31 is enabled to drive the AUXSIG1 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 30 | MUX30 | R/W | 0h | Selects the output of MUX30 to drive AUXSIG1 of CLB-XBAR<br>0: Respective output of MUX30 is disabled to drive the AUXSIG1 of CLB-XBAR<br>1: Respective output of MUX30 is enabled to drive the AUXSIG1 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29 | MUX29 | R/W | 0h | Selects the output of MUX29 to drive AUXSIG1 of CLB-XBAR<br>0: Respective output of MUX29 is disabled to drive the AUXSIG1 of CLB-XBAR<br>1: Respective output of MUX29 is enabled to drive the AUXSIG1 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 28 | MUX28 | R/W | 0h | Selects the output of MUX28 to drive AUXSIG1 of CLB-XBAR<br>0: Respective output of MUX28 is disabled to drive the AUXSIG1 of CLB-XBAR<br>1: Respective output of MUX28 is enabled to drive the AUXSIG1 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-78. AUXSIG1MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 27 | MUX27 | R/W | 0h | Selects the output of MUX27 to drive AUXSIG1 of CLB-XBAR<br>0: Respective output of MUX27 is disabled to drive the AUXSIG1 of CLB-XBAR<br>1: Respective output of MUX27 is enabled to drive the AUXSIG1 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 26 | MUX26 | R/W | 0h | Selects the output of MUX26 to drive AUXSIG1 of CLB-XBAR<br>0: Respective output of MUX26 is disabled to drive the AUXSIG1 of CLB-XBAR<br>1: Respective output of MUX26 is enabled to drive the AUXSIG1 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25 | MUX25 | R/W | 0h | Selects the output of MUX25 to drive AUXSIG1 of CLB-XBAR<br>0: Respective output of MUX25 is disabled to drive the AUXSIG1 of CLB-XBAR<br>1: Respective output of MUX25 is enabled to drive the AUXSIG1 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 24 | MUX24 | R/W | 0h | Selects the output of MUX24 to drive AUXSIG1 of CLB-XBAR<br>0: Respective output of MUX24 is disabled to drive the AUXSIG1 of CLB-XBAR<br>1: Respective output of MUX24 is enabled to drive the AUXSIG1 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23 | MUX23 | R/W | 0h | Selects the output of MUX23 to drive AUXSIG1 of CLB-XBAR<br>0: Respective output of MUX23 is disabled to drive the AUXSIG1 of CLB-XBAR<br>1: Respective output of MUX23 is enabled to drive the AUXSIG1 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 22 | MUX22 | R/W | 0h | Selects the output of MUX22 to drive AUXSIG1 of CLB-XBAR<br>0: Respective output of MUX22 is disabled to drive the AUXSIG1 of CLB-XBAR<br>1: Respective output of MUX22 is enabled to drive the AUXSIG1 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21 | MUX21 | R/W | 0h | Selects the output of MUX21 to drive AUXSIG1 of CLB-XBAR<br>0: Respective output of MUX21 is disabled to drive the AUXSIG1 of CLB-XBAR<br>1: Respective output of MUX21 is enabled to drive the AUXSIG1 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 20 | MUX20 | R/W | 0h | Selects the output of MUX20 to drive AUXSIG1 of CLB-XBAR<br>0: Respective output of MUX20 is disabled to drive the AUXSIG1 of CLB-XBAR<br>1: Respective output of MUX20 is enabled to drive the AUXSIG1 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-78. AUXSIG1MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19 | MUX19 | R/W | 0h | Selects the output of MUX19 to drive AUXSIG1 of CLB-XBAR<br>0: Respective output of MUX19 is disabled to drive the AUXSIG1 of CLB-XBAR<br>1: Respective output of MUX19 is enabled to drive the AUXSIG1 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 18 | MUX18 | R/W | 0h | Selects the output of MUX18 to drive AUXSIG1 of CLB-XBAR<br>0: Respective output of MUX18 is disabled to drive the AUXSIG1 of CLB-XBAR<br>1: Respective output of MUX18 is enabled to drive the AUXSIG1 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17 | MUX17 | R/W | 0h | Selects the output of MUX17 to drive AUXSIG1 of CLB-XBAR<br>0: Respective output of MUX17 is disabled to drive the AUXSIG1 of CLB-XBAR<br>1: Respective output of MUX17 is enabled to drive the AUXSIG1 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 16 | MUX16 | R/W | 0h | Selects the output of MUX16 to drive AUXSIG1 of CLB-XBAR<br>0: Respective output of MUX16 is disabled to drive the AUXSIG1 of CLB-XBAR<br>1: Respective output of MUX16 is enabled to drive the AUXSIG1 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15 | MUX15 | R/W | 0h | Selects the output of MUX15 to drive AUXSIG1 of CLB-XBAR<br>0: Respective output of MUX15 is disabled to drive the AUXSIG1 of CLB-XBAR<br>1: Respective output of MUX15 is enabled to drive the AUXSIG1 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 14 | MUX14 | R/W | 0h | Selects the output of MUX14 to drive AUXSIG1 of CLB-XBAR<br>0: Respective output of MUX14 is disabled to drive the AUXSIG1 of CLB-XBAR<br>1: Respective output of MUX14 is enabled to drive the AUXSIG1 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13 | MUX13 | R/W | 0h | Selects the output of MUX13 to drive AUXSIG1 of CLB-XBAR<br>0: Respective output of MUX13 is disabled to drive the AUXSIG1 of CLB-XBAR<br>1: Respective output of MUX13 is enabled to drive the AUXSIG1 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 12 | MUX12 | R/W | 0h | Selects the output of MUX12 to drive AUXSIG1 of CLB-XBAR<br>0: Respective output of MUX12 is disabled to drive the AUXSIG1 of CLB-XBAR<br>1: Respective output of MUX12 is enabled to drive the AUXSIG1 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-78. AUXSIG1MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 11 | MUX11 | R/W | 0h | Selects the output of MUX11 to drive AUXSIG1 of CLB-XBAR<br>0: Respective output of MUX11 is disabled to drive the AUXSIG1 of CLB-XBAR<br>1: Respective output of MUX11 is enabled to drive the AUXSIG1 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 10 | MUX10 | R/W | 0h | Selects the output of MUX10 to drive AUXSIG1 of CLB-XBAR<br>0: Respective output of MUX10 is disabled to drive the AUXSIG1 of CLB-XBAR<br>1: Respective output of MUX10 is enabled to drive the AUXSIG1 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9 | MUX9 | R/W | 0h | Selects the output of MUX9 to drive AUXSIG1 of CLB-XBAR<br>0: Respective output of MUX9 is disabled to drive the AUXSIG1 of CLB-XBAR<br>1: Respective output of MUX9 is enabled to drive the AUXSIG1 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 8 | MUX8 | R/W | 0h | Selects the output of MUX8 to drive AUXSIG1 of CLB-XBAR<br>0: Respective output of MUX8 is disabled to drive the AUXSIG1 of CLB-XBAR<br>1: Respective output of MUX8 is enabled to drive the AUXSIG1 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7 | MUX7 | R/W | 0h | Selects the output of MUX7 to drive AUXSIG1 of CLB-XBAR<br>0: Respective output of MUX7 is disabled to drive the AUXSIG1 of CLB-XBAR<br>1: Respective output of MUX7 is enabled to drive the AUXSIG1 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 6 | MUX6 | R/W | 0h | Selects the output of MUX6 to drive AUXSIG1 of CLB-XBAR<br>0: Respective output of MUX6 is disabled to drive the AUXSIG1 of CLB-XBAR<br>1: Respective output of MUX6 is enabled to drive the AUXSIG1 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5 | MUX5 | R/W | 0h | Selects the output of MUX5 to drive AUXSIG1 of CLB-XBAR<br>0: Respective output of MUX5 is disabled to drive the AUXSIG1 of CLB-XBAR<br>1: Respective output of MUX5 is enabled to drive the AUXSIG1 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 4 | MUX4 | R/W | 0h | Selects the output of MUX4 to drive AUXSIG1 of CLB-XBAR<br>0: Respective output of MUX4 is disabled to drive the AUXSIG1 of CLB-XBAR<br>1: Respective output of MUX4 is enabled to drive the AUXSIG1 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-78. AUXSIG1MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3 | MUX3 | R/W | 0h | Selects the output of MUX3 to drive AUXSIG1 of CLB-XBAR<br>0: Respective output of MUX3 is disabled to drive the AUXSIG1 of CLB-XBAR<br>1: Respective output of MUX3 is enabled to drive the AUXSIG1 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 2 | MUX2 | R/W | 0h | Selects the output of MUX2 to drive AUXSIG1 of CLB-XBAR<br>0: Respective output of MUX2 is disabled to drive the AUXSIG1 of CLB-XBAR<br>1: Respective output of MUX2 is enabled to drive the AUXSIG1 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1 | MUX1 | R/W | 0h | Selects the output of MUX1 to drive AUXSIG1 of CLB-XBAR<br>0: Respective output of MUX1 is disabled to drive the AUXSIG1 of CLB-XBAR<br>1: Respective output of MUX1 is enabled to drive the AUXSIG1 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 0 | MUX0 | R/W | 0h | Selects the output of mux0 to drive AUXSIG1 of CLB-XBAR<br>0: Respective output of MUX0 is disabled to drive the AUXSIG1 of CLB-XBAR<br>1: Respective output of MUX0 is enabled to drive the AUXSIG1 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.5.19 AUXSIG2MUXENABLE Register (Offset = 24h) [Reset = 00000000h]

AUXSIG2MUXENABLE is shown in Figure 9-72 and described in Table 9-79.

Return to the Summary Table.

CLB XBAR Mux Enable Register for Output-2

**Figure 9-72. AUXSIG2MUXENABLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MUX31 | MUX30 | MUX29 | MUX28 | MUX27 | MUX26 | MUX25 | MUX24 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MUX23 | MUX22 | MUX21 | MUX20 | MUX19 | MUX18 | MUX17 | MUX16 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MUX15 | MUX14 | MUX13 | MUX12 | MUX11 | MUX10 | MUX9 | MUX8 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | MUX6 | MUX5 | MUX4 | MUX3 | MUX2 | MUX1 | MUX0 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 9-79. AUXSIG2MUXENABLE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | MUX31 | R/W | 0h | Selects the output of MUX31 to drive AUXSIG2 of CLB-XBAR<br>0: Respective output of MUX31 is disabled to drive the AUXSIG2 of CLB-XBAR<br>1: Respective output of MUX31 is enabled to drive the AUXSIG2 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 30 | MUX30 | R/W | 0h | Selects the output of MUX30 to drive AUXSIG2 of CLB-XBAR<br>0: Respective output of MUX30 is disabled to drive the AUXSIG2 of CLB-XBAR<br>1: Respective output of MUX30 is enabled to drive the AUXSIG2 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29 | MUX29 | R/W | 0h | Selects the output of MUX29 to drive AUXSIG2 of CLB-XBAR<br>0: Respective output of MUX29 is disabled to drive the AUXSIG2 of CLB-XBAR<br>1: Respective output of MUX29 is enabled to drive the AUXSIG2 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 28 | MUX28 | R/W | 0h | Selects the output of MUX28 to drive AUXSIG2 of CLB-XBAR<br>0: Respective output of MUX28 is disabled to drive the AUXSIG2 of CLB-XBAR<br>1: Respective output of MUX28 is enabled to drive the AUXSIG2 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-79. AUXSIG2MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 27 | MUX27 | R/W | 0h | Selects the output of MUX27 to drive AUXSIG2 of CLB-XBAR<br>0: Respective output of MUX27 is disabled to drive the AUXSIG2 of CLB-XBAR<br>1: Respective output of MUX27 is enabled to drive the AUXSIG2 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 26 | MUX26 | R/W | 0h | Selects the output of MUX26 to drive AUXSIG2 of CLB-XBAR<br>0: Respective output of MUX26 is disabled to drive the AUXSIG2 of CLB-XBAR<br>1: Respective output of MUX26 is enabled to drive the AUXSIG2 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25 | MUX25 | R/W | 0h | Selects the output of MUX25 to drive AUXSIG2 of CLB-XBAR<br>0: Respective output of MUX25 is disabled to drive the AUXSIG2 of CLB-XBAR<br>1: Respective output of MUX25 is enabled to drive the AUXSIG2 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 24 | MUX24 | R/W | 0h | Selects the output of MUX24 to drive AUXSIG2 of CLB-XBAR<br>0: Respective output of MUX24 is disabled to drive the AUXSIG2 of CLB-XBAR<br>1: Respective output of MUX24 is enabled to drive the AUXSIG2 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23 | MUX23 | R/W | 0h | Selects the output of MUX23 to drive AUXSIG2 of CLB-XBAR<br>0: Respective output of MUX23 is disabled to drive the AUXSIG2 of CLB-XBAR<br>1: Respective output of MUX23 is enabled to drive the AUXSIG2 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 22 | MUX22 | R/W | 0h | Selects the output of MUX22 to drive AUXSIG2 of CLB-XBAR<br>0: Respective output of MUX22 is disabled to drive the AUXSIG2 of CLB-XBAR<br>1: Respective output of MUX22 is enabled to drive the AUXSIG2 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21 | MUX21 | R/W | 0h | Selects the output of MUX21 to drive AUXSIG2 of CLB-XBAR<br>0: Respective output of MUX21 is disabled to drive the AUXSIG2 of CLB-XBAR<br>1: Respective output of MUX21 is enabled to drive the AUXSIG2 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 20 | MUX20 | R/W | 0h | Selects the output of MUX20 to drive AUXSIG2 of CLB-XBAR<br>0: Respective output of MUX20 is disabled to drive the AUXSIG2 of CLB-XBAR<br>1: Respective output of MUX20 is enabled to drive the AUXSIG2 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-79. AUXSIG2MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19 | MUX19 | R/W | 0h | Selects the output of MUX19 to drive AUXSIG2 of CLB-XBAR<br>0: Respective output of MUX19 is disabled to drive the AUXSIG2 of CLB-XBAR<br>1: Respective output of MUX19 is enabled to drive the AUXSIG2 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 18 | MUX18 | R/W | 0h | Selects the output of MUX18 to drive AUXSIG2 of CLB-XBAR<br>0: Respective output of MUX18 is disabled to drive the AUXSIG2 of CLB-XBAR<br>1: Respective output of MUX18 is enabled to drive the AUXSIG2 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17 | MUX17 | R/W | 0h | Selects the output of MUX17 to drive AUXSIG2 of CLB-XBAR<br>0: Respective output of MUX17 is disabled to drive the AUXSIG2 of CLB-XBAR<br>1: Respective output of MUX17 is enabled to drive the AUXSIG2 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 16 | MUX16 | R/W | 0h | Selects the output of MUX16 to drive AUXSIG2 of CLB-XBAR<br>0: Respective output of MUX16 is disabled to drive the AUXSIG2 of CLB-XBAR<br>1: Respective output of MUX16 is enabled to drive the AUXSIG2 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15 | MUX15 | R/W | 0h | Selects the output of MUX15 to drive AUXSIG2 of CLB-XBAR<br>0: Respective output of MUX15 is disabled to drive the AUXSIG2 of CLB-XBAR<br>1: Respective output of MUX15 is enabled to drive the AUXSIG2 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 14 | MUX14 | R/W | 0h | Selects the output of MUX14 to drive AUXSIG2 of CLB-XBAR<br>0: Respective output of MUX14 is disabled to drive the AUXSIG2 of CLB-XBAR<br>1: Respective output of MUX14 is enabled to drive the AUXSIG2 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13 | MUX13 | R/W | 0h | Selects the output of MUX13 to drive AUXSIG2 of CLB-XBAR<br>0: Respective output of MUX13 is disabled to drive the AUXSIG2 of CLB-XBAR<br>1: Respective output of MUX13 is enabled to drive the AUXSIG2 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 12 | MUX12 | R/W | 0h | Selects the output of MUX12 to drive AUXSIG2 of CLB-XBAR<br>0: Respective output of MUX12 is disabled to drive the AUXSIG2 of CLB-XBAR<br>1: Respective output of MUX12 is enabled to drive the AUXSIG2 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-79. AUXSIG2MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 11 | MUX11 | R/W | 0h | Selects the output of MUX11 to drive AUXSIG2 of CLB-XBAR<br>0: Respective output of MUX11 is disabled to drive the AUXSIG2 of CLB-XBAR<br>1: Respective output of MUX11 is enabled to drive the AUXSIG2 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 10 | MUX10 | R/W | 0h | Selects the output of MUX10 to drive AUXSIG2 of CLB-XBAR<br>0: Respective output of MUX10 is disabled to drive the AUXSIG2 of CLB-XBAR<br>1: Respective output of MUX10 is enabled to drive the AUXSIG2 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9 | MUX9 | R/W | 0h | Selects the output of MUX9 to drive AUXSIG2 of CLB-XBAR<br>0: Respective output of MUX9 is disabled to drive the AUXSIG2 of CLB-XBAR<br>1: Respective output of MUX9 is enabled to drive the AUXSIG2 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 8 | MUX8 | R/W | 0h | Selects the output of MUX8 to drive AUXSIG2 of CLB-XBAR<br>0: Respective output of MUX8 is disabled to drive the AUXSIG2 of CLB-XBAR<br>1: Respective output of MUX8 is enabled to drive the AUXSIG2 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7 | MUX7 | R/W | 0h | Selects the output of MUX7 to drive AUXSIG2 of CLB-XBAR<br>0: Respective output of MUX7 is disabled to drive the AUXSIG2 of CLB-XBAR<br>1: Respective output of MUX7 is enabled to drive the AUXSIG2 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 6 | MUX6 | R/W | 0h | Selects the output of MUX6 to drive AUXSIG2 of CLB-XBAR<br>0: Respective output of MUX6 is disabled to drive the AUXSIG2 of CLB-XBAR<br>1: Respective output of MUX6 is enabled to drive the AUXSIG2 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5 | MUX5 | R/W | 0h | Selects the output of MUX5 to drive AUXSIG2 of CLB-XBAR<br>0: Respective output of MUX5 is disabled to drive the AUXSIG2 of CLB-XBAR<br>1: Respective output of MUX5 is enabled to drive the AUXSIG2 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 4 | MUX4 | R/W | 0h | Selects the output of MUX4 to drive AUXSIG2 of CLB-XBAR<br>0: Respective output of MUX4 is disabled to drive the AUXSIG2 of CLB-XBAR<br>1: Respective output of MUX4 is enabled to drive the AUXSIG2 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-79. AUXSIG2MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3 | MUX3 | R/W | 0h | Selects the output of MUX3 to drive AUXSIG2 of CLB-XBAR<br>0: Respective output of MUX3 is disabled to drive the AUXSIG2 of CLB-XBAR<br>1: Respective output of MUX3 is enabled to drive the AUXSIG2 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 2 | MUX2 | R/W | 0h | Selects the output of MUX2 to drive AUXSIG2 of CLB-XBAR<br>0: Respective output of MUX2 is disabled to drive the AUXSIG2 of CLB-XBAR<br>1: Respective output of MUX2 is enabled to drive the AUXSIG2 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1 | MUX1 | R/W | 0h | Selects the output of MUX1 to drive AUXSIG2 of CLB-XBAR<br>0: Respective output of MUX1 is disabled to drive the AUXSIG2 of CLB-XBAR<br>1: Respective output of MUX1 is enabled to drive the AUXSIG2 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 0 | MUX0 | R/W | 0h | Selects the output of mux0 to drive AUXSIG2 of CLB-XBAR<br>0: Respective output of MUX0 is disabled to drive the AUXSIG2 of CLB-XBAR<br>1: Respective output of MUX0 is enabled to drive the AUXSIG2 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.5.20 AUXSIG3MUXENABLE Register (Offset = 26h) [Reset = 00000000h]

AUXSIG3MUXENABLE is shown in Figure 9-73 and described in Table 9-80.

Return to the Summary Table.

CLB XBAR Mux Enable Register for Output-3

**Figure 9-73. AUXSIG3MUXENABLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MUX31 | MUX30 | MUX29 | MUX28 | MUX27 | MUX26 | MUX25 | MUX24 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MUX23 | MUX22 | MUX21 | MUX20 | MUX19 | MUX18 | MUX17 | MUX16 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MUX15 | MUX14 | MUX13 | MUX12 | MUX11 | MUX10 | MUX9 | MUX8 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | MUX6 | MUX5 | MUX4 | MUX3 | MUX2 | MUX1 | MUX0 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 9-80. AUXSIG3MUXENABLE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | MUX31 | R/W | 0h | Selects the output of MUX31 to drive AUXSIG3 of CLB-XBAR<br>0: Respective output of MUX31 is disabled to drive the AUXSIG3 of CLB-XBAR<br>1: Respective output of MUX31 is enabled to drive the AUXSIG3 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 30 | MUX30 | R/W | 0h | Selects the output of MUX30 to drive AUXSIG3 of CLB-XBAR<br>0: Respective output of MUX30 is disabled to drive the AUXSIG3 of CLB-XBAR<br>1: Respective output of MUX30 is enabled to drive the AUXSIG3 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29 | MUX29 | R/W | 0h | Selects the output of MUX29 to drive AUXSIG3 of CLB-XBAR<br>0: Respective output of MUX29 is disabled to drive the AUXSIG3 of CLB-XBAR<br>1: Respective output of MUX29 is enabled to drive the AUXSIG3 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 28 | MUX28 | R/W | 0h | Selects the output of MUX28 to drive AUXSIG3 of CLB-XBAR<br>0: Respective output of MUX28 is disabled to drive the AUXSIG3 of CLB-XBAR<br>1: Respective output of MUX28 is enabled to drive the AUXSIG3 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-80. AUXSIG3MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 27 | MUX27 | R/W | 0h | Selects the output of MUX27 to drive AUXSIG3 of CLB-XBAR<br>0: Respective output of MUX27 is disabled to drive the AUXSIG3 of CLB-XBAR<br>1: Respective output of MUX27 is enabled to drive the AUXSIG3 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 26 | MUX26 | R/W | 0h | Selects the output of MUX26 to drive AUXSIG3 of CLB-XBAR<br>0: Respective output of MUX26 is disabled to drive the AUXSIG3 of CLB-XBAR<br>1: Respective output of MUX26 is enabled to drive the AUXSIG3 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25 | MUX25 | R/W | 0h | Selects the output of MUX25 to drive AUXSIG3 of CLB-XBAR<br>0: Respective output of MUX25 is disabled to drive the AUXSIG3 of CLB-XBAR<br>1: Respective output of MUX25 is enabled to drive the AUXSIG3 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 24 | MUX24 | R/W | 0h | Selects the output of MUX24 to drive AUXSIG3 of CLB-XBAR<br>0: Respective output of MUX24 is disabled to drive the AUXSIG3 of CLB-XBAR<br>1: Respective output of MUX24 is enabled to drive the AUXSIG3 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23 | MUX23 | R/W | 0h | Selects the output of MUX23 to drive AUXSIG3 of CLB-XBAR<br>0: Respective output of MUX23 is disabled to drive the AUXSIG3 of CLB-XBAR<br>1: Respective output of MUX23 is enabled to drive the AUXSIG3 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 22 | MUX22 | R/W | 0h | Selects the output of MUX22 to drive AUXSIG3 of CLB-XBAR<br>0: Respective output of MUX22 is disabled to drive the AUXSIG3 of CLB-XBAR<br>1: Respective output of MUX22 is enabled to drive the AUXSIG3 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21 | MUX21 | R/W | 0h | Selects the output of MUX21 to drive AUXSIG3 of CLB-XBAR<br>0: Respective output of MUX21 is disabled to drive the AUXSIG3 of CLB-XBAR<br>1: Respective output of MUX21 is enabled to drive the AUXSIG3 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 20 | MUX20 | R/W | 0h | Selects the output of MUX20 to drive AUXSIG3 of CLB-XBAR<br>0: Respective output of MUX20 is disabled to drive the AUXSIG3 of CLB-XBAR<br>1: Respective output of MUX20 is enabled to drive the AUXSIG3 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-80. AUXSIG3MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19 | MUX19 | R/W | 0h | Selects the output of MUX19 to drive AUXSIG3 of CLB-XBAR<br>0: Respective output of MUX19 is disabled to drive the AUXSIG3 of CLB-XBAR<br>1: Respective output of MUX19 is enabled to drive the AUXSIG3 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 18 | MUX18 | R/W | 0h | Selects the output of MUX18 to drive AUXSIG3 of CLB-XBAR<br>0: Respective output of MUX18 is disabled to drive the AUXSIG3 of CLB-XBAR<br>1: Respective output of MUX18 is enabled to drive the AUXSIG3 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17 | MUX17 | R/W | 0h | Selects the output of MUX17 to drive AUXSIG3 of CLB-XBAR<br>0: Respective output of MUX17 is disabled to drive the AUXSIG3 of CLB-XBAR<br>1: Respective output of MUX17 is enabled to drive the AUXSIG3 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 16 | MUX16 | R/W | 0h | Selects the output of MUX16 to drive AUXSIG3 of CLB-XBAR<br>0: Respective output of MUX16 is disabled to drive the AUXSIG3 of CLB-XBAR<br>1: Respective output of MUX16 is enabled to drive the AUXSIG3 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15 | MUX15 | R/W | 0h | Selects the output of MUX15 to drive AUXSIG3 of CLB-XBAR<br>0: Respective output of MUX15 is disabled to drive the AUXSIG3 of CLB-XBAR<br>1: Respective output of MUX15 is enabled to drive the AUXSIG3 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 14 | MUX14 | R/W | 0h | Selects the output of MUX14 to drive AUXSIG3 of CLB-XBAR<br>0: Respective output of MUX14 is disabled to drive the AUXSIG3 of CLB-XBAR<br>1: Respective output of MUX14 is enabled to drive the AUXSIG3 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13 | MUX13 | R/W | 0h | Selects the output of MUX13 to drive AUXSIG3 of CLB-XBAR<br>0: Respective output of MUX13 is disabled to drive the AUXSIG3 of CLB-XBAR<br>1: Respective output of MUX13 is enabled to drive the AUXSIG3 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 12 | MUX12 | R/W | 0h | Selects the output of MUX12 to drive AUXSIG3 of CLB-XBAR<br>0: Respective output of MUX12 is disabled to drive the AUXSIG3 of CLB-XBAR<br>1: Respective output of MUX12 is enabled to drive the AUXSIG3 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-80. AUXSIG3MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 11 | MUX11 | R/W | 0h | Selects the output of MUX11 to drive AUXSIG3 of CLB-XBAR<br>0: Respective output of MUX11 is disabled to drive the AUXSIG3 of CLB-XBAR<br>1: Respective output of MUX11 is enabled to drive the AUXSIG3 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 10 | MUX10 | R/W | 0h | Selects the output of MUX10 to drive AUXSIG3 of CLB-XBAR<br>0: Respective output of MUX10 is disabled to drive the AUXSIG3 of CLB-XBAR<br>1: Respective output of MUX10 is enabled to drive the AUXSIG3 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9 | MUX9 | R/W | 0h | Selects the output of MUX9 to drive AUXSIG3 of CLB-XBAR<br>0: Respective output of MUX9 is disabled to drive the AUXSIG3 of CLB-XBAR<br>1: Respective output of MUX9 is enabled to drive the AUXSIG3 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 8 | MUX8 | R/W | 0h | Selects the output of MUX8 to drive AUXSIG3 of CLB-XBAR<br>0: Respective output of MUX8 is disabled to drive the AUXSIG3 of CLB-XBAR<br>1: Respective output of MUX8 is enabled to drive the AUXSIG3 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7 | MUX7 | R/W | 0h | Selects the output of MUX7 to drive AUXSIG3 of CLB-XBAR<br>0: Respective output of MUX7 is disabled to drive the AUXSIG3 of CLB-XBAR<br>1: Respective output of MUX7 is enabled to drive the AUXSIG3 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 6 | MUX6 | R/W | 0h | Selects the output of MUX6 to drive AUXSIG3 of CLB-XBAR<br>0: Respective output of MUX6 is disabled to drive the AUXSIG3 of CLB-XBAR<br>1: Respective output of MUX6 is enabled to drive the AUXSIG3 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5 | MUX5 | R/W | 0h | Selects the output of MUX5 to drive AUXSIG3 of CLB-XBAR<br>0: Respective output of MUX5 is disabled to drive the AUXSIG3 of CLB-XBAR<br>1: Respective output of MUX5 is enabled to drive the AUXSIG3 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 4 | MUX4 | R/W | 0h | Selects the output of MUX4 to drive AUXSIG3 of CLB-XBAR<br>0: Respective output of MUX4 is disabled to drive the AUXSIG3 of CLB-XBAR<br>1: Respective output of MUX4 is enabled to drive the AUXSIG3 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-80. AUXSIG3MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3 | MUX3 | R/W | 0h | Selects the output of MUX3 to drive AUXSIG3 of CLB-XBAR<br>0: Respective output of MUX3 is disabled to drive the AUXSIG3 of CLB-XBAR<br>1: Respective output of MUX3 is enabled to drive the AUXSIG3 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 2 | MUX2 | R/W | 0h | Selects the output of MUX2 to drive AUXSIG3 of CLB-XBAR<br>0: Respective output of MUX2 is disabled to drive the AUXSIG3 of CLB-XBAR<br>1: Respective output of MUX2 is enabled to drive the AUXSIG3 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1 | MUX1 | R/W | 0h | Selects the output of MUX1 to drive AUXSIG3 of CLB-XBAR<br>0: Respective output of MUX1 is disabled to drive the AUXSIG3 of CLB-XBAR<br>1: Respective output of MUX1 is enabled to drive the AUXSIG3 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 0 | MUX0 | R/W | 0h | Selects the output of mux0 to drive AUXSIG3 of CLB-XBAR<br>0: Respective output of MUX0 is disabled to drive the AUXSIG3 of CLB-XBAR<br>1: Respective output of MUX0 is enabled to drive the AUXSIG3 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.5.21 AUXSIG4MUXENABLE Register (Offset = 28h) [Reset = 00000000h]

AUXSIG4MUXENABLE is shown in Figure 9-74 and described in Table 9-81.

Return to the Summary Table.

CLB XBAR Mux Enable Register for Output-4

**Figure 9-74. AUXSIG4MUXENABLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MUX31 | MUX30 | MUX29 | MUX28 | MUX27 | MUX26 | MUX25 | MUX24 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MUX23 | MUX22 | MUX21 | MUX20 | MUX19 | MUX18 | MUX17 | MUX16 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MUX15 | MUX14 | MUX13 | MUX12 | MUX11 | MUX10 | MUX9 | MUX8 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | MUX6 | MUX5 | MUX4 | MUX3 | MUX2 | MUX1 | MUX0 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 9-81. AUXSIG4MUXENABLE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | MUX31 | R/W | 0h | Selects the output of MUX31 to drive AUXSIG4 of CLB-XBAR<br>0: Respective output of MUX31 is disabled to drive the AUXSIG4 of CLB-XBAR<br>1: Respective output of MUX31 is enabled to drive the AUXSIG4 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 30 | MUX30 | R/W | 0h | Selects the output of MUX30 to drive AUXSIG4 of CLB-XBAR<br>0: Respective output of MUX30 is disabled to drive the AUXSIG4 of CLB-XBAR<br>1: Respective output of MUX30 is enabled to drive the AUXSIG4 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29 | MUX29 | R/W | 0h | Selects the output of MUX29 to drive AUXSIG4 of CLB-XBAR<br>0: Respective output of MUX29 is disabled to drive the AUXSIG4 of CLB-XBAR<br>1: Respective output of MUX29 is enabled to drive the AUXSIG4 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 28 | MUX28 | R/W | 0h | Selects the output of MUX28 to drive AUXSIG4 of CLB-XBAR<br>0: Respective output of MUX28 is disabled to drive the AUXSIG4 of CLB-XBAR<br>1: Respective output of MUX28 is enabled to drive the AUXSIG4 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-81. AUXSIG4MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 27 | MUX27 | R/W | 0h | Selects the output of MUX27 to drive AUXSIG4 of CLB-XBAR<br>0: Respective output of MUX27 is disabled to drive the AUXSIG4 of CLB-XBAR<br>1: Respective output of MUX27 is enabled to drive the AUXSIG4 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 26 | MUX26 | R/W | 0h | Selects the output of MUX26 to drive AUXSIG4 of CLB-XBAR<br>0: Respective output of MUX26 is disabled to drive the AUXSIG4 of CLB-XBAR<br>1: Respective output of MUX26 is enabled to drive the AUXSIG4 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25 | MUX25 | R/W | 0h | Selects the output of MUX25 to drive AUXSIG4 of CLB-XBAR<br>0: Respective output of MUX25 is disabled to drive the AUXSIG4 of CLB-XBAR<br>1: Respective output of MUX25 is enabled to drive the AUXSIG4 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 24 | MUX24 | R/W | 0h | Selects the output of MUX24 to drive AUXSIG4 of CLB-XBAR<br>0: Respective output of MUX24 is disabled to drive the AUXSIG4 of CLB-XBAR<br>1: Respective output of MUX24 is enabled to drive the AUXSIG4 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23 | MUX23 | R/W | 0h | Selects the output of MUX23 to drive AUXSIG4 of CLB-XBAR<br>0: Respective output of MUX23 is disabled to drive the AUXSIG4 of CLB-XBAR<br>1: Respective output of MUX23 is enabled to drive the AUXSIG4 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 22 | MUX22 | R/W | 0h | Selects the output of MUX22 to drive AUXSIG4 of CLB-XBAR<br>0: Respective output of MUX22 is disabled to drive the AUXSIG4 of CLB-XBAR<br>1: Respective output of MUX22 is enabled to drive the AUXSIG4 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21 | MUX21 | R/W | 0h | Selects the output of MUX21 to drive AUXSIG4 of CLB-XBAR<br>0: Respective output of MUX21 is disabled to drive the AUXSIG4 of CLB-XBAR<br>1: Respective output of MUX21 is enabled to drive the AUXSIG4 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 20 | MUX20 | R/W | 0h | Selects the output of MUX20 to drive AUXSIG4 of CLB-XBAR<br>0: Respective output of MUX20 is disabled to drive the AUXSIG4 of CLB-XBAR<br>1: Respective output of MUX20 is enabled to drive the AUXSIG4 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-81. AUXSIG4MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19 | MUX19 | R/W | 0h | Selects the output of MUX19 to drive AUXSIG4 of CLB-XBAR<br>0: Respective output of MUX19 is disabled to drive the AUXSIG4 of CLB-XBAR<br>1: Respective output of MUX19 is enabled to drive the AUXSIG4 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 18 | MUX18 | R/W | 0h | Selects the output of MUX18 to drive AUXSIG4 of CLB-XBAR<br>0: Respective output of MUX18 is disabled to drive the AUXSIG4 of CLB-XBAR<br>1: Respective output of MUX18 is enabled to drive the AUXSIG4 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17 | MUX17 | R/W | 0h | Selects the output of MUX17 to drive AUXSIG4 of CLB-XBAR<br>0: Respective output of MUX17 is disabled to drive the AUXSIG4 of CLB-XBAR<br>1: Respective output of MUX17 is enabled to drive the AUXSIG4 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 16 | MUX16 | R/W | 0h | Selects the output of MUX16 to drive AUXSIG4 of CLB-XBAR<br>0: Respective output of MUX16 is disabled to drive the AUXSIG4 of CLB-XBAR<br>1: Respective output of MUX16 is enabled to drive the AUXSIG4 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15 | MUX15 | R/W | 0h | Selects the output of MUX15 to drive AUXSIG4 of CLB-XBAR<br>0: Respective output of MUX15 is disabled to drive the AUXSIG4 of CLB-XBAR<br>1: Respective output of MUX15 is enabled to drive the AUXSIG4 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 14 | MUX14 | R/W | 0h | Selects the output of MUX14 to drive AUXSIG4 of CLB-XBAR<br>0: Respective output of MUX14 is disabled to drive the AUXSIG4 of CLB-XBAR<br>1: Respective output of MUX14 is enabled to drive the AUXSIG4 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13 | MUX13 | R/W | 0h | Selects the output of MUX13 to drive AUXSIG4 of CLB-XBAR<br>0: Respective output of MUX13 is disabled to drive the AUXSIG4 of CLB-XBAR<br>1: Respective output of MUX13 is enabled to drive the AUXSIG4 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 12 | MUX12 | R/W | 0h | Selects the output of MUX12 to drive AUXSIG4 of CLB-XBAR<br>0: Respective output of MUX12 is disabled to drive the AUXSIG4 of CLB-XBAR<br>1: Respective output of MUX12 is enabled to drive the AUXSIG4 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-81. AUXSIG4MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 11 | MUX11 | R/W | 0h | Selects the output of MUX11 to drive AUXSIG4 of CLB-XBAR<br>0: Respective output of MUX11 is disabled to drive the AUXSIG4 of CLB-XBAR<br>1: Respective output of MUX11 is enabled to drive the AUXSIG4 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 10 | MUX10 | R/W | 0h | Selects the output of MUX10 to drive AUXSIG4 of CLB-XBAR<br>0: Respective output of MUX10 is disabled to drive the AUXSIG4 of CLB-XBAR<br>1: Respective output of MUX10 is enabled to drive the AUXSIG4 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9 | MUX9 | R/W | 0h | Selects the output of MUX9 to drive AUXSIG4 of CLB-XBAR<br>0: Respective output of MUX9 is disabled to drive the AUXSIG4 of CLB-XBAR<br>1: Respective output of MUX9 is enabled to drive the AUXSIG4 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 8 | MUX8 | R/W | 0h | Selects the output of MUX8 to drive AUXSIG4 of CLB-XBAR<br>0: Respective output of MUX8 is disabled to drive the AUXSIG4 of CLB-XBAR<br>1: Respective output of MUX8 is enabled to drive the AUXSIG4 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7 | MUX7 | R/W | 0h | Selects the output of MUX7 to drive AUXSIG4 of CLB-XBAR<br>0: Respective output of MUX7 is disabled to drive the AUXSIG4 of CLB-XBAR<br>1: Respective output of MUX7 is enabled to drive the AUXSIG4 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 6 | MUX6 | R/W | 0h | Selects the output of MUX6 to drive AUXSIG4 of CLB-XBAR<br>0: Respective output of MUX6 is disabled to drive the AUXSIG4 of CLB-XBAR<br>1: Respective output of MUX6 is enabled to drive the AUXSIG4 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5 | MUX5 | R/W | 0h | Selects the output of MUX5 to drive AUXSIG4 of CLB-XBAR<br>0: Respective output of MUX5 is disabled to drive the AUXSIG4 of CLB-XBAR<br>1: Respective output of MUX5 is enabled to drive the AUXSIG4 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 4 | MUX4 | R/W | 0h | Selects the output of MUX4 to drive AUXSIG4 of CLB-XBAR<br>0: Respective output of MUX4 is disabled to drive the AUXSIG4 of CLB-XBAR<br>1: Respective output of MUX4 is enabled to drive the AUXSIG4 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-81. AUXSIG4MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3 | MUX3 | R/W | 0h | Selects the output of MUX3 to drive AUXSIG4 of CLB-XBAR<br>0: Respective output of MUX3 is disabled to drive the AUXSIG4 of CLB-XBAR<br>1: Respective output of MUX3 is enabled to drive the AUXSIG4 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 2 | MUX2 | R/W | 0h | Selects the output of MUX2 to drive AUXSIG4 of CLB-XBAR<br>0: Respective output of MUX2 is disabled to drive the AUXSIG4 of CLB-XBAR<br>1: Respective output of MUX2 is enabled to drive the AUXSIG4 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1 | MUX1 | R/W | 0h | Selects the output of MUX1 to drive AUXSIG4 of CLB-XBAR<br>0: Respective output of MUX1 is disabled to drive the AUXSIG4 of CLB-XBAR<br>1: Respective output of MUX1 is enabled to drive the AUXSIG4 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 0 | MUX0 | R/W | 0h | Selects the output of mux0 to drive AUXSIG4 of CLB-XBAR<br>0: Respective output of MUX0 is disabled to drive the AUXSIG4 of CLB-XBAR<br>1: Respective output of MUX0 is enabled to drive the AUXSIG4 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.5.22 AUXSIG5MUXENABLE Register (Offset = 2Ah) [Reset = 00000000h]

AUXSIG5MUXENABLE is shown in Figure 9-75 and described in Table 9-82.

Return to the Summary Table.

CLB XBAR Mux Enable Register for Output-5

**Figure 9-75. AUXSIG5MUXENABLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MUX31 | MUX30 | MUX29 | MUX28 | MUX27 | MUX26 | MUX25 | MUX24 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MUX23 | MUX22 | MUX21 | MUX20 | MUX19 | MUX18 | MUX17 | MUX16 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MUX15 | MUX14 | MUX13 | MUX12 | MUX11 | MUX10 | MUX9 | MUX8 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | MUX6 | MUX5 | MUX4 | MUX3 | MUX2 | MUX1 | MUX0 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 9-82. AUXSIG5MUXENABLE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | MUX31 | R/W | 0h | Selects the output of MUX31 to drive AUXSIG5 of CLB-XBAR<br>0: Respective output of MUX31 is disabled to drive the AUXSIG5 of CLB-XBAR<br>1: Respective output of MUX31 is enabled to drive the AUXSIG5 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 30 | MUX30 | R/W | 0h | Selects the output of MUX30 to drive AUXSIG5 of CLB-XBAR<br>0: Respective output of MUX30 is disabled to drive the AUXSIG5 of CLB-XBAR<br>1: Respective output of MUX30 is enabled to drive the AUXSIG5 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29 | MUX29 | R/W | 0h | Selects the output of MUX29 to drive AUXSIG5 of CLB-XBAR<br>0: Respective output of MUX29 is disabled to drive the AUXSIG5 of CLB-XBAR<br>1: Respective output of MUX29 is enabled to drive the AUXSIG5 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 28 | MUX28 | R/W | 0h | Selects the output of MUX28 to drive AUXSIG5 of CLB-XBAR<br>0: Respective output of MUX28 is disabled to drive the AUXSIG5 of CLB-XBAR<br>1: Respective output of MUX28 is enabled to drive the AUXSIG5 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-82. AUXSIG5MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 27 | MUX27 | R/W | 0h | Selects the output of MUX27 to drive AUXSIG5 of CLB-XBAR<br>0: Respective output of MUX27 is disabled to drive the AUXSIG5 of CLB-XBAR<br>1: Respective output of MUX27 is enabled to drive the AUXSIG5 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 26 | MUX26 | R/W | 0h | Selects the output of MUX26 to drive AUXSIG5 of CLB-XBAR<br>0: Respective output of MUX26 is disabled to drive the AUXSIG5 of CLB-XBAR<br>1: Respective output of MUX26 is enabled to drive the AUXSIG5 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25 | MUX25 | R/W | 0h | Selects the output of MUX25 to drive AUXSIG5 of CLB-XBAR<br>0: Respective output of MUX25 is disabled to drive the AUXSIG5 of CLB-XBAR<br>1: Respective output of MUX25 is enabled to drive the AUXSIG5 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 24 | MUX24 | R/W | 0h | Selects the output of MUX24 to drive AUXSIG5 of CLB-XBAR<br>0: Respective output of MUX24 is disabled to drive the AUXSIG5 of CLB-XBAR<br>1: Respective output of MUX24 is enabled to drive the AUXSIG5 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23 | MUX23 | R/W | 0h | Selects the output of MUX23 to drive AUXSIG5 of CLB-XBAR<br>0: Respective output of MUX23 is disabled to drive the AUXSIG5 of CLB-XBAR<br>1: Respective output of MUX23 is enabled to drive the AUXSIG5 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 22 | MUX22 | R/W | 0h | Selects the output of MUX22 to drive AUXSIG5 of CLB-XBAR<br>0: Respective output of MUX22 is disabled to drive the AUXSIG5 of CLB-XBAR<br>1: Respective output of MUX22 is enabled to drive the AUXSIG5 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21 | MUX21 | R/W | 0h | Selects the output of MUX21 to drive AUXSIG5 of CLB-XBAR<br>0: Respective output of MUX21 is disabled to drive the AUXSIG5 of CLB-XBAR<br>1: Respective output of MUX21 is enabled to drive the AUXSIG5 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 20 | MUX20 | R/W | 0h | Selects the output of MUX20 to drive AUXSIG5 of CLB-XBAR<br>0: Respective output of MUX20 is disabled to drive the AUXSIG5 of CLB-XBAR<br>1: Respective output of MUX20 is enabled to drive the AUXSIG5 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-82. AUXSIG5MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19 | MUX19 | R/W | 0h | Selects the output of MUX19 to drive AUXSIG5 of CLB-XBAR<br>0: Respective output of MUX19 is disabled to drive the AUXSIG5 of CLB-XBAR<br>1: Respective output of MUX19 is enabled to drive the AUXSIG5 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 18 | MUX18 | R/W | 0h | Selects the output of MUX18 to drive AUXSIG5 of CLB-XBAR<br>0: Respective output of MUX18 is disabled to drive the AUXSIG5 of CLB-XBAR<br>1: Respective output of MUX18 is enabled to drive the AUXSIG5 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17 | MUX17 | R/W | 0h | Selects the output of MUX17 to drive AUXSIG5 of CLB-XBAR<br>0: Respective output of MUX17 is disabled to drive the AUXSIG5 of CLB-XBAR<br>1: Respective output of MUX17 is enabled to drive the AUXSIG5 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 16 | MUX16 | R/W | 0h | Selects the output of MUX16 to drive AUXSIG5 of CLB-XBAR<br>0: Respective output of MUX16 is disabled to drive the AUXSIG5 of CLB-XBAR<br>1: Respective output of MUX16 is enabled to drive the AUXSIG5 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15 | MUX15 | R/W | 0h | Selects the output of MUX15 to drive AUXSIG5 of CLB-XBAR<br>0: Respective output of MUX15 is disabled to drive the AUXSIG5 of CLB-XBAR<br>1: Respective output of MUX15 is enabled to drive the AUXSIG5 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 14 | MUX14 | R/W | 0h | Selects the output of MUX14 to drive AUXSIG5 of CLB-XBAR<br>0: Respective output of MUX14 is disabled to drive the AUXSIG5 of CLB-XBAR<br>1: Respective output of MUX14 is enabled to drive the AUXSIG5 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13 | MUX13 | R/W | 0h | Selects the output of MUX13 to drive AUXSIG5 of CLB-XBAR<br>0: Respective output of MUX13 is disabled to drive the AUXSIG5 of CLB-XBAR<br>1: Respective output of MUX13 is enabled to drive the AUXSIG5 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 12 | MUX12 | R/W | 0h | Selects the output of MUX12 to drive AUXSIG5 of CLB-XBAR<br>0: Respective output of MUX12 is disabled to drive the AUXSIG5 of CLB-XBAR<br>1: Respective output of MUX12 is enabled to drive the AUXSIG5 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-82. AUXSIG5MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 11 | MUX11 | R/W | 0h | Selects the output of MUX11 to drive AUXSIG5 of CLB-XBAR<br>0: Respective output of MUX11 is disabled to drive the AUXSIG5 of CLB-XBAR<br>1: Respective output of MUX11 is enabled to drive the AUXSIG5 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 10 | MUX10 | R/W | 0h | Selects the output of MUX10 to drive AUXSIG5 of CLB-XBAR<br>0: Respective output of MUX10 is disabled to drive the AUXSIG5 of CLB-XBAR<br>1: Respective output of MUX10 is enabled to drive the AUXSIG5 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9 | MUX9 | R/W | 0h | Selects the output of MUX9 to drive AUXSIG5 of CLB-XBAR<br>0: Respective output of MUX9 is disabled to drive the AUXSIG5 of CLB-XBAR<br>1: Respective output of MUX9 is enabled to drive the AUXSIG5 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 8 | MUX8 | R/W | 0h | Selects the output of MUX8 to drive AUXSIG5 of CLB-XBAR<br>0: Respective output of MUX8 is disabled to drive the AUXSIG5 of CLB-XBAR<br>1: Respective output of MUX8 is enabled to drive the AUXSIG5 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7 | MUX7 | R/W | 0h | Selects the output of MUX7 to drive AUXSIG5 of CLB-XBAR<br>0: Respective output of MUX7 is disabled to drive the AUXSIG5 of CLB-XBAR<br>1: Respective output of MUX7 is enabled to drive the AUXSIG5 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 6 | MUX6 | R/W | 0h | Selects the output of MUX6 to drive AUXSIG5 of CLB-XBAR<br>0: Respective output of MUX6 is disabled to drive the AUXSIG5 of CLB-XBAR<br>1: Respective output of MUX6 is enabled to drive the AUXSIG5 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5 | MUX5 | R/W | 0h | Selects the output of MUX5 to drive AUXSIG5 of CLB-XBAR<br>0: Respective output of MUX5 is disabled to drive the AUXSIG5 of CLB-XBAR<br>1: Respective output of MUX5 is enabled to drive the AUXSIG5 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 4 | MUX4 | R/W | 0h | Selects the output of MUX4 to drive AUXSIG5 of CLB-XBAR<br>0: Respective output of MUX4 is disabled to drive the AUXSIG5 of CLB-XBAR<br>1: Respective output of MUX4 is enabled to drive the AUXSIG5 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-82. AUXSIG5MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3 | MUX3 | R/W | 0h | Selects the output of MUX3 to drive AUXSIG5 of CLB-XBAR<br>0: Respective output of MUX3 is disabled to drive the AUXSIG5 of CLB-XBAR<br>1: Respective output of MUX3 is enabled to drive the AUXSIG5 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 2 | MUX2 | R/W | 0h | Selects the output of MUX2 to drive AUXSIG5 of CLB-XBAR<br>0: Respective output of MUX2 is disabled to drive the AUXSIG5 of CLB-XBAR<br>1: Respective output of MUX2 is enabled to drive the AUXSIG5 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1 | MUX1 | R/W | 0h | Selects the output of MUX1 to drive AUXSIG5 of CLB-XBAR<br>0: Respective output of MUX1 is disabled to drive the AUXSIG5 of CLB-XBAR<br>1: Respective output of MUX1 is enabled to drive the AUXSIG5 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 0 | MUX0 | R/W | 0h | Selects the output of mux0 to drive AUXSIG5 of CLB-XBAR<br>0: Respective output of MUX0 is disabled to drive the AUXSIG5 of CLB-XBAR<br>1: Respective output of MUX0 is enabled to drive the AUXSIG5 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.5.23 AUXSIG6MUXENABLE Register (Offset = 2Ch) [Reset = 00000000h]

AUXSIG6MUXENABLE is shown in Figure 9-76 and described in Table 9-83.

Return to the Summary Table.

CLB XBAR Mux Enable Register for Output-6

**Figure 9-76. AUXSIG6MUXENABLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MUX31 | MUX30 | MUX29 | MUX28 | MUX27 | MUX26 | MUX25 | MUX24 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MUX23 | MUX22 | MUX21 | MUX20 | MUX19 | MUX18 | MUX17 | MUX16 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MUX15 | MUX14 | MUX13 | MUX12 | MUX11 | MUX10 | MUX9 | MUX8 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | MUX6 | MUX5 | MUX4 | MUX3 | MUX2 | MUX1 | MUX0 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 9-83. AUXSIG6MUXENABLE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | MUX31 | R/W | 0h | Selects the output of MUX31 to drive AUXSIG6 of CLB-XBAR<br>0: Respective output of MUX31 is disabled to drive the AUXSIG6 of CLB-XBAR<br>1: Respective output of MUX31 is enabled to drive the AUXSIG6 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 30 | MUX30 | R/W | 0h | Selects the output of MUX30 to drive AUXSIG6 of CLB-XBAR<br>0: Respective output of MUX30 is disabled to drive the AUXSIG6 of CLB-XBAR<br>1: Respective output of MUX30 is enabled to drive the AUXSIG6 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29 | MUX29 | R/W | 0h | Selects the output of MUX29 to drive AUXSIG6 of CLB-XBAR<br>0: Respective output of MUX29 is disabled to drive the AUXSIG6 of CLB-XBAR<br>1: Respective output of MUX29 is enabled to drive the AUXSIG6 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 28 | MUX28 | R/W | 0h | Selects the output of MUX28 to drive AUXSIG6 of CLB-XBAR<br>0: Respective output of MUX28 is disabled to drive the AUXSIG6 of CLB-XBAR<br>1: Respective output of MUX28 is enabled to drive the AUXSIG6 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-83. AUXSIG6MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 27 | MUX27 | R/W | 0h | Selects the output of MUX27 to drive AUXSIG6 of CLB-XBAR<br>0: Respective output of MUX27 is disabled to drive the AUXSIG6 of CLB-XBAR<br>1: Respective output of MUX27 is enabled to drive the AUXSIG6 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 26 | MUX26 | R/W | 0h | Selects the output of MUX26 to drive AUXSIG6 of CLB-XBAR<br>0: Respective output of MUX26 is disabled to drive the AUXSIG6 of CLB-XBAR<br>1: Respective output of MUX26 is enabled to drive the AUXSIG6 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25 | MUX25 | R/W | 0h | Selects the output of MUX25 to drive AUXSIG6 of CLB-XBAR<br>0: Respective output of MUX25 is disabled to drive the AUXSIG6 of CLB-XBAR<br>1: Respective output of MUX25 is enabled to drive the AUXSIG6 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 24 | MUX24 | R/W | 0h | Selects the output of MUX24 to drive AUXSIG6 of CLB-XBAR<br>0: Respective output of MUX24 is disabled to drive the AUXSIG6 of CLB-XBAR<br>1: Respective output of MUX24 is enabled to drive the AUXSIG6 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23 | MUX23 | R/W | 0h | Selects the output of MUX23 to drive AUXSIG6 of CLB-XBAR<br>0: Respective output of MUX23 is disabled to drive the AUXSIG6 of CLB-XBAR<br>1: Respective output of MUX23 is enabled to drive the AUXSIG6 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 22 | MUX22 | R/W | 0h | Selects the output of MUX22 to drive AUXSIG6 of CLB-XBAR<br>0: Respective output of MUX22 is disabled to drive the AUXSIG6 of CLB-XBAR<br>1: Respective output of MUX22 is enabled to drive the AUXSIG6 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21 | MUX21 | R/W | 0h | Selects the output of MUX21 to drive AUXSIG6 of CLB-XBAR<br>0: Respective output of MUX21 is disabled to drive the AUXSIG6 of CLB-XBAR<br>1: Respective output of MUX21 is enabled to drive the AUXSIG6 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 20 | MUX20 | R/W | 0h | Selects the output of MUX20 to drive AUXSIG6 of CLB-XBAR<br>0: Respective output of MUX20 is disabled to drive the AUXSIG6 of CLB-XBAR<br>1: Respective output of MUX20 is enabled to drive the AUXSIG6 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-83. AUXSIG6MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19 | MUX19 | R/W | 0h | Selects the output of MUX19 to drive AUXSIG6 of CLB-XBAR<br>0: Respective output of MUX19 is disabled to drive the AUXSIG6 of CLB-XBAR<br>1: Respective output of MUX19 is enabled to drive the AUXSIG6 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 18 | MUX18 | R/W | 0h | Selects the output of MUX18 to drive AUXSIG6 of CLB-XBAR<br>0: Respective output of MUX18 is disabled to drive the AUXSIG6 of CLB-XBAR<br>1: Respective output of MUX18 is enabled to drive the AUXSIG6 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17 | MUX17 | R/W | 0h | Selects the output of MUX17 to drive AUXSIG6 of CLB-XBAR<br>0: Respective output of MUX17 is disabled to drive the AUXSIG6 of CLB-XBAR<br>1: Respective output of MUX17 is enabled to drive the AUXSIG6 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 16 | MUX16 | R/W | 0h | Selects the output of MUX16 to drive AUXSIG6 of CLB-XBAR<br>0: Respective output of MUX16 is disabled to drive the AUXSIG6 of CLB-XBAR<br>1: Respective output of MUX16 is enabled to drive the AUXSIG6 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15 | MUX15 | R/W | 0h | Selects the output of MUX15 to drive AUXSIG6 of CLB-XBAR<br>0: Respective output of MUX15 is disabled to drive the AUXSIG6 of CLB-XBAR<br>1: Respective output of MUX15 is enabled to drive the AUXSIG6 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 14 | MUX14 | R/W | 0h | Selects the output of MUX14 to drive AUXSIG6 of CLB-XBAR<br>0: Respective output of MUX14 is disabled to drive the AUXSIG6 of CLB-XBAR<br>1: Respective output of MUX14 is enabled to drive the AUXSIG6 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13 | MUX13 | R/W | 0h | Selects the output of MUX13 to drive AUXSIG6 of CLB-XBAR<br>0: Respective output of MUX13 is disabled to drive the AUXSIG6 of CLB-XBAR<br>1: Respective output of MUX13 is enabled to drive the AUXSIG6 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 12 | MUX12 | R/W | 0h | Selects the output of MUX12 to drive AUXSIG6 of CLB-XBAR<br>0: Respective output of MUX12 is disabled to drive the AUXSIG6 of CLB-XBAR<br>1: Respective output of MUX12 is enabled to drive the AUXSIG6 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-83. AUXSIG6MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 11 | MUX11 | R/W | 0h | Selects the output of MUX11 to drive AUXSIG6 of CLB-XBAR<br>0: Respective output of MUX11 is disabled to drive the AUXSIG6 of CLB-XBAR<br>1: Respective output of MUX11 is enabled to drive the AUXSIG6 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 10 | MUX10 | R/W | 0h | Selects the output of MUX10 to drive AUXSIG6 of CLB-XBAR<br>0: Respective output of MUX10 is disabled to drive the AUXSIG6 of CLB-XBAR<br>1: Respective output of MUX10 is enabled to drive the AUXSIG6 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9 | MUX9 | R/W | 0h | Selects the output of MUX9 to drive AUXSIG6 of CLB-XBAR<br>0: Respective output of MUX9 is disabled to drive the AUXSIG6 of CLB-XBAR<br>1: Respective output of MUX9 is enabled to drive the AUXSIG6 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 8 | MUX8 | R/W | 0h | Selects the output of MUX8 to drive AUXSIG6 of CLB-XBAR<br>0: Respective output of MUX8 is disabled to drive the AUXSIG6 of CLB-XBAR<br>1: Respective output of MUX8 is enabled to drive the AUXSIG6 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7 | MUX7 | R/W | 0h | Selects the output of MUX7 to drive AUXSIG6 of CLB-XBAR<br>0: Respective output of MUX7 is disabled to drive the AUXSIG6 of CLB-XBAR<br>1: Respective output of MUX7 is enabled to drive the AUXSIG6 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 6 | MUX6 | R/W | 0h | Selects the output of MUX6 to drive AUXSIG6 of CLB-XBAR<br>0: Respective output of MUX6 is disabled to drive the AUXSIG6 of CLB-XBAR<br>1: Respective output of MUX6 is enabled to drive the AUXSIG6 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5 | MUX5 | R/W | 0h | Selects the output of MUX5 to drive AUXSIG6 of CLB-XBAR<br>0: Respective output of MUX5 is disabled to drive the AUXSIG6 of CLB-XBAR<br>1: Respective output of MUX5 is enabled to drive the AUXSIG6 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 4 | MUX4 | R/W | 0h | Selects the output of MUX4 to drive AUXSIG6 of CLB-XBAR<br>0: Respective output of MUX4 is disabled to drive the AUXSIG6 of CLB-XBAR<br>1: Respective output of MUX4 is enabled to drive the AUXSIG6 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-83. AUXSIG6MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3 | MUX3 | R/W | 0h | Selects the output of MUX3 to drive AUXSIG6 of CLB-XBAR<br>0: Respective output of MUX3 is disabled to drive the AUXSIG6 of CLB-XBAR<br>1: Respective output of MUX3 is enabled to drive the AUXSIG6 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 2 | MUX2 | R/W | 0h | Selects the output of MUX2 to drive AUXSIG6 of CLB-XBAR<br>0: Respective output of MUX2 is disabled to drive the AUXSIG6 of CLB-XBAR<br>1: Respective output of MUX2 is enabled to drive the AUXSIG6 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1 | MUX1 | R/W | 0h | Selects the output of MUX1 to drive AUXSIG6 of CLB-XBAR<br>0: Respective output of MUX1 is disabled to drive the AUXSIG6 of CLB-XBAR<br>1: Respective output of MUX1 is enabled to drive the AUXSIG6 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 0 | MUX0 | R/W | 0h | Selects the output of mux0 to drive AUXSIG6 of CLB-XBAR<br>0: Respective output of MUX0 is disabled to drive the AUXSIG6 of CLB-XBAR<br>1: Respective output of MUX0 is enabled to drive the AUXSIG6 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.5.24 AUXSIG7MUXENABLE Register (Offset = 2Eh) [Reset = 00000000h]

AUXSIG7MUXENABLE is shown in Figure 9-77 and described in Table 9-84.

Return to the Summary Table.

CLB XBAR Mux Enable Register for Output-7

**Figure 9-77. AUXSIG7MUXENABLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MUX31 | MUX30 | MUX29 | MUX28 | MUX27 | MUX26 | MUX25 | MUX24 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MUX23 | MUX22 | MUX21 | MUX20 | MUX19 | MUX18 | MUX17 | MUX16 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MUX15 | MUX14 | MUX13 | MUX12 | MUX11 | MUX10 | MUX9 | MUX8 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | MUX6 | MUX5 | MUX4 | MUX3 | MUX2 | MUX1 | MUX0 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 9-84. AUXSIG7MUXENABLE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | MUX31 | R/W | 0h | Selects the output of MUX31 to drive AUXSIG7 of CLB-XBAR<br>0: Respective output of MUX31 is disabled to drive the AUXSIG7 of CLB-XBAR<br>1: Respective output of MUX31 is enabled to drive the AUXSIG7 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 30 | MUX30 | R/W | 0h | Selects the output of MUX30 to drive AUXSIG7 of CLB-XBAR<br>0: Respective output of MUX30 is disabled to drive the AUXSIG7 of CLB-XBAR<br>1: Respective output of MUX30 is enabled to drive the AUXSIG7 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29 | MUX29 | R/W | 0h | Selects the output of MUX29 to drive AUXSIG7 of CLB-XBAR<br>0: Respective output of MUX29 is disabled to drive the AUXSIG7 of CLB-XBAR<br>1: Respective output of MUX29 is enabled to drive the AUXSIG7 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 28 | MUX28 | R/W | 0h | Selects the output of MUX28 to drive AUXSIG7 of CLB-XBAR<br>0: Respective output of MUX28 is disabled to drive the AUXSIG7 of CLB-XBAR<br>1: Respective output of MUX28 is enabled to drive the AUXSIG7 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-84. AUXSIG7MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 27 | MUX27 | R/W | 0h | Selects the output of MUX27 to drive AUXSIG7 of CLB-XBAR<br>0: Respective output of MUX27 is disabled to drive the AUXSIG7 of CLB-XBAR<br>1: Respective output of MUX27 is enabled to drive the AUXSIG7 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 26 | MUX26 | R/W | 0h | Selects the output of MUX26 to drive AUXSIG7 of CLB-XBAR<br>0: Respective output of MUX26 is disabled to drive the AUXSIG7 of CLB-XBAR<br>1: Respective output of MUX26 is enabled to drive the AUXSIG7 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25 | MUX25 | R/W | 0h | Selects the output of MUX25 to drive AUXSIG7 of CLB-XBAR<br>0: Respective output of MUX25 is disabled to drive the AUXSIG7 of CLB-XBAR<br>1: Respective output of MUX25 is enabled to drive the AUXSIG7 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 24 | MUX24 | R/W | 0h | Selects the output of MUX24 to drive AUXSIG7 of CLB-XBAR<br>0: Respective output of MUX24 is disabled to drive the AUXSIG7 of CLB-XBAR<br>1: Respective output of MUX24 is enabled to drive the AUXSIG7 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23 | MUX23 | R/W | 0h | Selects the output of MUX23 to drive AUXSIG7 of CLB-XBAR<br>0: Respective output of MUX23 is disabled to drive the AUXSIG7 of CLB-XBAR<br>1: Respective output of MUX23 is enabled to drive the AUXSIG7 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 22 | MUX22 | R/W | 0h | Selects the output of MUX22 to drive AUXSIG7 of CLB-XBAR<br>0: Respective output of MUX22 is disabled to drive the AUXSIG7 of CLB-XBAR<br>1: Respective output of MUX22 is enabled to drive the AUXSIG7 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21 | MUX21 | R/W | 0h | Selects the output of MUX21 to drive AUXSIG7 of CLB-XBAR<br>0: Respective output of MUX21 is disabled to drive the AUXSIG7 of CLB-XBAR<br>1: Respective output of MUX21 is enabled to drive the AUXSIG7 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 20 | MUX20 | R/W | 0h | Selects the output of MUX20 to drive AUXSIG7 of CLB-XBAR<br>0: Respective output of MUX20 is disabled to drive the AUXSIG7 of CLB-XBAR<br>1: Respective output of MUX20 is enabled to drive the AUXSIG7 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-84. AUXSIG7MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19 | MUX19 | R/W | 0h | Selects the output of MUX19 to drive AUXSIG7 of CLB-XBAR<br>0: Respective output of MUX19 is disabled to drive the AUXSIG7 of CLB-XBAR<br>1: Respective output of MUX19 is enabled to drive the AUXSIG7 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 18 | MUX18 | R/W | 0h | Selects the output of MUX18 to drive AUXSIG7 of CLB-XBAR<br>0: Respective output of MUX18 is disabled to drive the AUXSIG7 of CLB-XBAR<br>1: Respective output of MUX18 is enabled to drive the AUXSIG7 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17 | MUX17 | R/W | 0h | Selects the output of MUX17 to drive AUXSIG7 of CLB-XBAR<br>0: Respective output of MUX17 is disabled to drive the AUXSIG7 of CLB-XBAR<br>1: Respective output of MUX17 is enabled to drive the AUXSIG7 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 16 | MUX16 | R/W | 0h | Selects the output of MUX16 to drive AUXSIG7 of CLB-XBAR<br>0: Respective output of MUX16 is disabled to drive the AUXSIG7 of CLB-XBAR<br>1: Respective output of MUX16 is enabled to drive the AUXSIG7 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15 | MUX15 | R/W | 0h | Selects the output of MUX15 to drive AUXSIG7 of CLB-XBAR<br>0: Respective output of MUX15 is disabled to drive the AUXSIG7 of CLB-XBAR<br>1: Respective output of MUX15 is enabled to drive the AUXSIG7 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 14 | MUX14 | R/W | 0h | Selects the output of MUX14 to drive AUXSIG7 of CLB-XBAR<br>0: Respective output of MUX14 is disabled to drive the AUXSIG7 of CLB-XBAR<br>1: Respective output of MUX14 is enabled to drive the AUXSIG7 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13 | MUX13 | R/W | 0h | Selects the output of MUX13 to drive AUXSIG7 of CLB-XBAR<br>0: Respective output of MUX13 is disabled to drive the AUXSIG7 of CLB-XBAR<br>1: Respective output of MUX13 is enabled to drive the AUXSIG7 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 12 | MUX12 | R/W | 0h | Selects the output of MUX12 to drive AUXSIG7 of CLB-XBAR<br>0: Respective output of MUX12 is disabled to drive the AUXSIG7 of CLB-XBAR<br>1: Respective output of MUX12 is enabled to drive the AUXSIG7 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-84. AUXSIG7MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 11 | MUX11 | R/W | 0h | Selects the output of MUX11 to drive AUXSIG7 of CLB-XBAR<br>0: Respective output of MUX11 is disabled to drive the AUXSIG7 of CLB-XBAR<br>1: Respective output of MUX11 is enabled to drive the AUXSIG7 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 10 | MUX10 | R/W | 0h | Selects the output of MUX10 to drive AUXSIG7 of CLB-XBAR<br>0: Respective output of MUX10 is disabled to drive the AUXSIG7 of CLB-XBAR<br>1: Respective output of MUX10 is enabled to drive the AUXSIG7 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9 | MUX9 | R/W | 0h | Selects the output of MUX9 to drive AUXSIG7 of CLB-XBAR<br>0: Respective output of MUX9 is disabled to drive the AUXSIG7 of CLB-XBAR<br>1: Respective output of MUX9 is enabled to drive the AUXSIG7 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 8 | MUX8 | R/W | 0h | Selects the output of MUX8 to drive AUXSIG7 of CLB-XBAR<br>0: Respective output of MUX8 is disabled to drive the AUXSIG7 of CLB-XBAR<br>1: Respective output of MUX8 is enabled to drive the AUXSIG7 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7 | MUX7 | R/W | 0h | Selects the output of MUX7 to drive AUXSIG7 of CLB-XBAR<br>0: Respective output of MUX7 is disabled to drive the AUXSIG7 of CLB-XBAR<br>1: Respective output of MUX7 is enabled to drive the AUXSIG7 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 6 | MUX6 | R/W | 0h | Selects the output of MUX6 to drive AUXSIG7 of CLB-XBAR<br>0: Respective output of MUX6 is disabled to drive the AUXSIG7 of CLB-XBAR<br>1: Respective output of MUX6 is enabled to drive the AUXSIG7 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5 | MUX5 | R/W | 0h | Selects the output of MUX5 to drive AUXSIG7 of CLB-XBAR<br>0: Respective output of MUX5 is disabled to drive the AUXSIG7 of CLB-XBAR<br>1: Respective output of MUX5 is enabled to drive the AUXSIG7 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 4 | MUX4 | R/W | 0h | Selects the output of MUX4 to drive AUXSIG7 of CLB-XBAR<br>0: Respective output of MUX4 is disabled to drive the AUXSIG7 of CLB-XBAR<br>1: Respective output of MUX4 is enabled to drive the AUXSIG7 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-84. AUXSIG7MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3 | MUX3 | R/W | 0h | Selects the output of MUX3 to drive AUXSIG7 of CLB-XBAR<br>0: Respective output of MUX3 is disabled to drive the AUXSIG7 of CLB-XBAR<br>1: Respective output of MUX3 is enabled to drive the AUXSIG7 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 2 | MUX2 | R/W | 0h | Selects the output of MUX2 to drive AUXSIG7 of CLB-XBAR<br>0: Respective output of MUX2 is disabled to drive the AUXSIG7 of CLB-XBAR<br>1: Respective output of MUX2 is enabled to drive the AUXSIG7 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1 | MUX1 | R/W | 0h | Selects the output of MUX1 to drive AUXSIG7 of CLB-XBAR<br>0: Respective output of MUX1 is disabled to drive the AUXSIG7 of CLB-XBAR<br>1: Respective output of MUX1 is enabled to drive the AUXSIG7 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 0 | MUX0 | R/W | 0h | Selects the output of mux0 to drive AUXSIG7 of CLB-XBAR<br>0: Respective output of MUX0 is disabled to drive the AUXSIG7 of CLB-XBAR<br>1: Respective output of MUX0 is enabled to drive the AUXSIG7 of CLB-XBAR<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.5.25 AUXSIGOUTINV Register (Offset = 38h) [Reset = 00000000h]

AUXSIGOUTINV is shown in Figure 9-78 and described in Table 9-85.

Return to the Summary Table.

CLB XBAR Output Inversion Register

**Figure 9-78. AUXSIGOUTINV Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OUT7 | OUT6 | OUT5 | OUT4 | OUT3 | OUT2 | OUT1 | OUT0 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 9-85. AUXSIGOUTINV Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-8 | RESERVED | R-0 | 0h | Reserved |
| 7 | OUT7 | R/W | 0h | Selects polarity for AUXSIG7 of CLB-XBAR<br>0: drives active high output<br>1: drives active-low output<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 6 | OUT6 | R/W | 0h | Selects polarity for AUXSIG6 of CLB-XBAR<br>0: drives active high output<br>1: drives active-low output<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5 | OUT5 | R/W | 0h | Selects polarity for AUXSIG5 of CLB-XBAR<br>0: drives active high output<br>1: drives active-low output<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 4 | OUT4 | R/W | 0h | Selects polarity for AUXSIG4 of CLB-XBAR<br>0: drives active high output<br>1: drives active-low output<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 3 | OUT3 | R/W | 0h | Selects polarity for AUXSIG3 of CLB-XBAR<br>0: drives active high output<br>1: drives active-low output<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 2 | OUT2 | R/W | 0h | Selects polarity for AUXSIG2 of CLB-XBAR<br>0: drives active high output<br>1: drives active-low output<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-85. AUXSIGOUTINV Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1 | OUT1 | R/W | 0h | Selects polarity for AUXSIG1 of CLB-XBAR<br>0: drives active high output<br>1: drives active-low output<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 0 | OUT0 | R/W | 0h | Selects polarity for AUXSIG0 of CLB-XBAR<br>0: drives active high output<br>1: drives active-low output<br>Refer to the CLB X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.5.26 AUXSIGLOCK Register (Offset = 3Eh) [Reset = 00000000h]

AUXSIGLOCK is shown in Figure 9-79 and described in Table 9-86.

Return to the Summary Table.

ClbXbar Configuration Lock register

**Figure 9-79. AUXSIGLOCK Register**

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

**Table 9-86. AUXSIGLOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | KEY | R-0/W | 0h | Bit-0 of this register can be set only if KEY= 0x5a5a<br>Reset type: CPU1.SYSRSn |
| 15-1 | RESERVED | R-0 | 0h | Reserved |
| 0 | LOCK | R/WSonce | 0h | Locks the configuration for CLB-XBAR. Once the configuration is locked, writes to the below registers for CLB-XBAR is blocked.<br>Registers Affected by the LOCK mechanism:<br>CLB-XBAROUTyMUX0TO15CFG<br>CLB-XBAROUTyMUX16TO31CFG<br>CLB-XBAROUTyMUXENABLE<br>CLB-XBAROUTLATEN<br>CLB-XBAROUTINV<br>0: Writes to the above registers are allowed<br>1: Writes to the above registers are blocked<br>Note:<br>[1] LOCK mechanism only apples to writes. Reads are never blocked.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.6 OUTPUT_XBAR_REGS Registers

Table 9-87 lists the memory-mapped registers for the OUTPUT_XBAR_REGS registers. All register offset addresses not listed in Table 9-87 should be considered as reserved locations and the register contents should not be modified.

**Table 9-87. OUTPUT_XBAR_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | OUTPUT1MUX0TO15CFG | Output X-BAR Mux Configuration for Output 1 | EALLOW | Go |
| 2h | OUTPUT1MUX16TO31CFG | Output X-BAR Mux Configuration for Output 1 | EALLOW | Go |
| 4h | OUTPUT2MUX0TO15CFG | Output X-BAR Mux Configuration for Output 2 | EALLOW | Go |
| 6h | OUTPUT2MUX16TO31CFG | Output X-BAR Mux Configuration for Output 2 | EALLOW | Go |
| 8h | OUTPUT3MUX0TO15CFG | Output X-BAR Mux Configuration for Output 3 | EALLOW | Go |
| Ah | OUTPUT3MUX16TO31CFG | Output X-BAR Mux Configuration for Output 3 | EALLOW | Go |
| Ch | OUTPUT4MUX0TO15CFG | Output X-BAR Mux Configuration for Output 4 | EALLOW | Go |
| Eh | OUTPUT4MUX16TO31CFG | Output X-BAR Mux Configuration for Output 4 | EALLOW | Go |
| 10h | OUTPUT5MUX0TO15CFG | Output X-BAR Mux Configuration for Output 5 | EALLOW | Go |
| 12h | OUTPUT5MUX16TO31CFG | Output X-BAR Mux Configuration for Output 5 | EALLOW | Go |
| 14h | OUTPUT6MUX0TO15CFG | Output X-BAR Mux Configuration for Output 6 | EALLOW | Go |
| 16h | OUTPUT6MUX16TO31CFG | Output X-BAR Mux Configuration for Output 6 | EALLOW | Go |
| 18h | OUTPUT7MUX0TO15CFG | Output X-BAR Mux Configuration for Output 7 | EALLOW | Go |
| 1Ah | OUTPUT7MUX16TO31CFG | Output X-BAR Mux Configuration for Output 7 | EALLOW | Go |
| 1Ch | OUTPUT8MUX0TO15CFG | Output X-BAR Mux Configuration for Output 8 | EALLOW | Go |
| 1Eh | OUTPUT8MUX16TO31CFG | Output X-BAR Mux Configuration for Output 8 | EALLOW | Go |
| 20h | OUTPUT1MUXENABLE | Output X-BAR Mux Enable for Output 1 | EALLOW | Go |
| 22h | OUTPUT2MUXENABLE | Output X-BAR Mux Enable for Output 2 | EALLOW | Go |
| 24h | OUTPUT3MUXENABLE | Output X-BAR Mux Enable for Output 3 | EALLOW | Go |
| 26h | OUTPUT4MUXENABLE | Output X-BAR Mux Enable for Output 4 | EALLOW | Go |
| 28h | OUTPUT5MUXENABLE | Output X-BAR Mux Enable for Output 5 | EALLOW | Go |
| 2Ah | OUTPUT6MUXENABLE | Output X-BAR Mux Enable for Output 6 | EALLOW | Go |
| 2Ch | OUTPUT7MUXENABLE | Output X-BAR Mux Enable for Output 7 | EALLOW | Go |
| 2Eh | OUTPUT8MUXENABLE | Output X-BAR Mux Enable for Output 8 | EALLOW | Go |
| 30h | OUTPUTLATCH | Output X-BAR Output Latch | | Go |
| 32h | OUTPUTLATCHCLR | Output X-BAR Output Latch Clear | | Go |
| 34h | OUTPUTLATCHFRC | Output X-BAR Output Latch Clear | | Go |
| 36h | OUTPUTLATCHENABLE | Output X-BAR Output Latch Enable | EALLOW | Go |
| 38h | OUTPUTINV | Output X-BAR Output Inversion | EALLOW | Go |
| 3Eh | OUTPUTLOCK | Output X-BAR Configuration Lock register | EALLOW | Go |

Complex bit access types are encoded to fit into small table cells. Table 9-88 shows the codes that are used for access types in this section.

**Table 9-88. OUTPUT_XBAR_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| R-0 | R<br>-0 | Read<br>Returns 0s |
| Write Type | | |
| W | W | Write |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-88. OUTPUT_XBAR_REGS Access Type Codes (continued)**

| Access Type | Code | Description |
|---|---|---|
| W1S | W<br>1S | Write<br>1 to set |
| WSonce | W<br>Sonce | Write<br>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. |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.6.1 OUTPUT1MUX0TO15CFG Register (Offset = 0h) [Reset = 00000000h]

OUTPUT1MUX0TO15CFG is shown in Figure 9-80 and described in Table 9-89.

Return to the Summary Table.

Output X-BAR Mux Configuration for Output 1

**Figure 9-80. 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 9-89. OUTPUT1MUX0TO15CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX15 | R/W | 0h | Select Bits for OUTPUT1 Mux15:<br>00 : Select .0 input for Mux15<br>01 : Select .1 input for Mux15<br>10 : Select .2 input for Mux15<br>11 : Select .3 input for Mux15<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX14 | R/W | 0h | Select Bits for OUTPUT1 Mux14:<br>00 : Select .0 input for Mux14<br>01 : Select .1 input for Mux14<br>10 : Select .2 input for Mux14<br>11 : Select .3 input for Mux14<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX13 | R/W | 0h | Select Bits for OUTPUT1 Mux13:<br>00 : Select .0 input for Mux13<br>01 : Select .1 input for Mux13<br>10 : Select .2 input for Mux13<br>11 : Select .3 input for Mux13<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX12 | R/W | 0h | Select Bits for OUTPUT1 Mux12:<br>00 : Select .0 input for Mux12<br>01 : Select .1 input for Mux12<br>10 : Select .2 input for Mux12<br>11 : Select .3 input for Mux12<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX11 | R/W | 0h | Select Bits for OUTPUT1 Mux11:<br>00 : Select .0 input for Mux11<br>01 : Select .1 input for Mux11<br>10 : Select .2 input for Mux11<br>11 : Select .3 input for Mux11<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21-20 | MUX10 | R/W | 0h | Select Bits for OUTPUT1 Mux10:<br>00 : Select .0 input for Mux10<br>01 : Select .1 input for Mux10<br>10 : Select .2 input for Mux10<br>11 : Select .3 input for Mux10<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-89. OUTPUT1MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19-18 | MUX9 | R/W | 0h | Select Bits for OUTPUT1 Mux9:<br>00 : Select .0 input for Mux9<br>01 : Select .1 input for Mux9<br>10 : Select .2 input for Mux9<br>11 : Select .3 input for Mux9<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX8 | R/W | 0h | Select Bits for OUTPUT1 Mux8:<br>00 : Select .0 input for Mux8<br>01 : Select .1 input for Mux8<br>10 : Select .2 input for Mux8<br>11 : Select .3 input for Mux8<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX7 | R/W | 0h | Select Bits for OUTPUT1 Mux7:<br>00 : Select .0 input for Mux7<br>01 : Select .1 input for Mux7<br>10 : Select .2 input for Mux7<br>11 : Select .3 input for Mux7<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX6 | R/W | 0h | Select Bits for OUTPUT1 Mux6:<br>00 : Select .0 input for Mux6<br>01 : Select .1 input for Mux6<br>10 : Select .2 input for Mux6<br>11 : Select .3 input for Mux6<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX5 | R/W | 0h | Select Bits for OUTPUT1 Mux5:<br>00 : Select .0 input for Mux5<br>01 : Select .1 input for Mux5<br>10 : Select .2 input for Mux5<br>11 : Select .3 input for Mux5<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX4 | R/W | 0h | Select Bits for OUTPUT1 Mux4:<br>00 : Select .0 input for Mux4<br>01 : Select .1 input for Mux4<br>10 : Select .2 input for Mux4<br>11 : Select .3 input for Mux4<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7-6 | MUX3 | R/W | 0h | Select Bits for OUTPUT1 Mux3:<br>00 : Select .0 input for Mux3<br>01 : Select .1 input for Mux3<br>10 : Select .2 input for Mux3<br>11 : Select .3 input for Mux3<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX2 | R/W | 0h | Select Bits for OUTPUT1 Mux2:<br>00 : Select .0 input for Mux2<br>01 : Select .1 input for Mux2<br>10 : Select .2 input for Mux2<br>11 : Select .3 input for Mux2<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-89. OUTPUT1MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-2 | MUX1 | R/W | 0h | Select Bits for OUTPUT1 Mux1:<br>00 : Select .0 input for Mux1<br>01 : Select .1 input for Mux1<br>10 : Select .2 input for Mux1<br>11 : Select .3 input for Mux1<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX0 | R/W | 0h | Select Bits for OUTPUT1 Mux0:<br>00 : Select .0 input for Mux0<br>01 : Select .1 input for Mux0<br>10 : Select .2 input for Mux0<br>11 : Select .3 input for Mux0<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.6.2 OUTPUT1MUX16TO31CFG Register (Offset = 2h) [Reset = 00000000h]

OUTPUT1MUX16TO31CFG is shown in Figure 9-81 and described in Table 9-90.

Return to the Summary Table.

Output X-BAR Mux Configuration for Output 1

**Figure 9-81. 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 9-90. OUTPUT1MUX16TO31CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX31 | R/W | 0h | Select Bits for OUTPUT1 Mux31:<br>00 : Select .0 input for Mux31<br>01 : Select .1 input for Mux31<br>10 : Select .2 input for Mux31<br>11 : Select .3 input for Mux31<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX30 | R/W | 0h | Select Bits for OUTPUT1 Mux30:<br>00 : Select .0 input for Mux30<br>01 : Select .1 input for Mux30<br>10 : Select .2 input for Mux30<br>11 : Select .3 input for Mux30<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX29 | R/W | 0h | Select Bits for OUTPUT1 Mux29:<br>00 : Select .0 input for Mux29<br>01 : Select .1 input for Mux29<br>10 : Select .2 input for Mux29<br>11 : Select .3 input for Mux29<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX28 | R/W | 0h | Select Bits for OUTPUT1 Mux28:<br>00 : Select .0 input for Mux28<br>01 : Select .1 input for Mux28<br>10 : Select .2 input for Mux28<br>11 : Select .3 input for Mux28<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX27 | R/W | 0h | Select Bits for OUTPUT1 Mux27:<br>00 : Select .0 input for Mux27<br>01 : Select .1 input for Mux27<br>10 : Select .2 input for Mux27<br>11 : Select .3 input for Mux27<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21-20 | MUX26 | R/W | 0h | Select Bits for OUTPUT1 Mux26:<br>00 : Select .0 input for Mux26<br>01 : Select .1 input for Mux26<br>10 : Select .2 input for Mux26<br>11 : Select .3 input for Mux26<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-90. OUTPUT1MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19-18 | MUX25 | R/W | 0h | Select Bits for OUTPUT1 Mux25:<br>00 : Select .0 input for Mux25<br>01 : Select .1 input for Mux25<br>10 : Select .2 input for Mux25<br>11 : Select .3 input for Mux25<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX24 | R/W | 0h | Select Bits for OUTPUT1 Mux24:<br>00 : Select .0 input for Mux24<br>01 : Select .1 input for Mux24<br>10 : Select .2 input for Mux24<br>11 : Select .3 input for Mux24<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX23 | R/W | 0h | Select Bits for OUTPUT1 Mux23:<br>00 : Select .0 input for Mux23<br>01 : Select .1 input for Mux23<br>10 : Select .2 input for Mux23<br>11 : Select .3 input for Mux23<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX22 | R/W | 0h | Select Bits for OUTPUT1 Mux22:<br>00 : Select .0 input for Mux22<br>01 : Select .1 input for Mux22<br>10 : Select .2 input for Mux22<br>11 : Select .3 input for Mux22<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX21 | R/W | 0h | Select Bits for OUTPUT1 Mux21:<br>00 : Select .0 input for Mux21<br>01 : Select .1 input for Mux21<br>10 : Select .2 input for Mux21<br>11 : Select .3 input for Mux21<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX20 | R/W | 0h | Select Bits for OUTPUT1 Mux20:<br>00 : Select .0 input for Mux20<br>01 : Select .1 input for Mux20<br>10 : Select .2 input for Mux20<br>11 : Select .3 input for Mux20<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7-6 | MUX19 | R/W | 0h | Select Bits for OUTPUT1 Mux19:<br>00 : Select .0 input for Mux19<br>01 : Select .1 input for Mux19<br>10 : Select .2 input for Mux19<br>11 : Select .3 input for Mux19<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX18 | R/W | 0h | Select Bits for OUTPUT1 Mux18:<br>00 : Select .0 input for Mux18<br>01 : Select .1 input for Mux18<br>10 : Select .2 input for Mux18<br>11 : Select .3 input for Mux18<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-90. OUTPUT1MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-2 | MUX17 | R/W | 0h | Select Bits for OUTPUT1 Mux17:<br>00 : Select .0 input for Mux17<br>01 : Select .1 input for Mux17<br>10 : Select .2 input for Mux17<br>11 : Select .3 input for Mux17<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX16 | R/W | 0h | Select Bits for OUTPUT1 Mux16:<br>00 : Select .0 input for Mux16<br>01 : Select .1 input for Mux16<br>10 : Select .2 input for Mux16<br>11 : Select .3 input for Mux16<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.6.3 OUTPUT2MUX0TO15CFG Register (Offset = 4h) [Reset = 00000000h]

OUTPUT2MUX0TO15CFG is shown in Figure 9-82 and described in Table 9-91.

Return to the Summary Table.

Output X-BAR Mux Configuration for Output 2

**Figure 9-82. 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 9-91. OUTPUT2MUX0TO15CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX15 | R/W | 0h | Select Bits for OUTPUT2 Mux15:<br>00 : Select .0 input for Mux15<br>01 : Select .1 input for Mux15<br>10 : Select .2 input for Mux15<br>11 : Select .3 input for Mux15<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX14 | R/W | 0h | Select Bits for OUTPUT2 Mux14:<br>00 : Select .0 input for Mux14<br>01 : Select .1 input for Mux14<br>10 : Select .2 input for Mux14<br>11 : Select .3 input for Mux14<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX13 | R/W | 0h | Select Bits for OUTPUT2 Mux13:<br>00 : Select .0 input for Mux13<br>01 : Select .1 input for Mux13<br>10 : Select .2 input for Mux13<br>11 : Select .3 input for Mux13<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX12 | R/W | 0h | Select Bits for OUTPUT2 Mux12:<br>00 : Select .0 input for Mux12<br>01 : Select .1 input for Mux12<br>10 : Select .2 input for Mux12<br>11 : Select .3 input for Mux12<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX11 | R/W | 0h | Select Bits for OUTPUT2 Mux11:<br>00 : Select .0 input for Mux11<br>01 : Select .1 input for Mux11<br>10 : Select .2 input for Mux11<br>11 : Select .3 input for Mux11<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21-20 | MUX10 | R/W | 0h | Select Bits for OUTPUT2 Mux10:<br>00 : Select .0 input for Mux10<br>01 : Select .1 input for Mux10<br>10 : Select .2 input for Mux10<br>11 : Select .3 input for Mux10<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-91. OUTPUT2MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19-18 | MUX9 | R/W | 0h | Select Bits for OUTPUT2 Mux9:<br>00 : Select .0 input for Mux9<br>01 : Select .1 input for Mux9<br>10 : Select .2 input for Mux9<br>11 : Select .3 input for Mux9<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX8 | R/W | 0h | Select Bits for OUTPUT2 Mux8:<br>00 : Select .0 input for Mux8<br>01 : Select .1 input for Mux8<br>10 : Select .2 input for Mux8<br>11 : Select .3 input for Mux8<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX7 | R/W | 0h | Select Bits for OUTPUT2 Mux7:<br>00 : Select .0 input for Mux7<br>01 : Select .1 input for Mux7<br>10 : Select .2 input for Mux7<br>11 : Select .3 input for Mux7<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX6 | R/W | 0h | Select Bits for OUTPUT2 Mux6:<br>00 : Select .0 input for Mux6<br>01 : Select .1 input for Mux6<br>10 : Select .2 input for Mux6<br>11 : Select .3 input for Mux6<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX5 | R/W | 0h | Select Bits for OUTPUT2 Mux5:<br>00 : Select .0 input for Mux5<br>01 : Select .1 input for Mux5<br>10 : Select .2 input for Mux5<br>11 : Select .3 input for Mux5<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX4 | R/W | 0h | Select Bits for OUTPUT2 Mux4:<br>00 : Select .0 input for Mux4<br>01 : Select .1 input for Mux4<br>10 : Select .2 input for Mux4<br>11 : Select .3 input for Mux4<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7-6 | MUX3 | R/W | 0h | Select Bits for OUTPUT2 Mux3:<br>00 : Select .0 input for Mux3<br>01 : Select .1 input for Mux3<br>10 : Select .2 input for Mux3<br>11 : Select .3 input for Mux3<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX2 | R/W | 0h | Select Bits for OUTPUT2 Mux2:<br>00 : Select .0 input for Mux2<br>01 : Select .1 input for Mux2<br>10 : Select .2 input for Mux2<br>11 : Select .3 input for Mux2<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-91. OUTPUT2MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-2 | MUX1 | R/W | 0h | Select Bits for OUTPUT2 Mux1:<br>00 : Select .0 input for Mux1<br>01 : Select .1 input for Mux1<br>10 : Select .2 input for Mux1<br>11 : Select .3 input for Mux1<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX0 | R/W | 0h | Select Bits for OUTPUT2 Mux0:<br>00 : Select .0 input for Mux0<br>01 : Select .1 input for Mux0<br>10 : Select .2 input for Mux0<br>11 : Select .3 input for Mux0<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.6.4 OUTPUT2MUX16TO31CFG Register (Offset = 6h) [Reset = 00000000h]

OUTPUT2MUX16TO31CFG is shown in Figure 9-83 and described in Table 9-92.

Return to the Summary Table.

Output X-BAR Mux Configuration for Output 2

**Figure 9-83. 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 9-92. OUTPUT2MUX16TO31CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX31 | R/W | 0h | Select Bits for OUTPUT2 Mux31:<br>00 : Select .0 input for Mux31<br>01 : Select .1 input for Mux31<br>10 : Select .2 input for Mux31<br>11 : Select .3 input for Mux31<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX30 | R/W | 0h | Select Bits for OUTPUT2 Mux30:<br>00 : Select .0 input for Mux30<br>01 : Select .1 input for Mux30<br>10 : Select .2 input for Mux30<br>11 : Select .3 input for Mux30<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX29 | R/W | 0h | Select Bits for OUTPUT2 Mux29:<br>00 : Select .0 input for Mux29<br>01 : Select .1 input for Mux29<br>10 : Select .2 input for Mux29<br>11 : Select .3 input for Mux29<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX28 | R/W | 0h | Select Bits for OUTPUT2 Mux28:<br>00 : Select .0 input for Mux28<br>01 : Select .1 input for Mux28<br>10 : Select .2 input for Mux28<br>11 : Select .3 input for Mux28<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX27 | R/W | 0h | Select Bits for OUTPUT2 Mux27:<br>00 : Select .0 input for Mux27<br>01 : Select .1 input for Mux27<br>10 : Select .2 input for Mux27<br>11 : Select .3 input for Mux27<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21-20 | MUX26 | R/W | 0h | Select Bits for OUTPUT2 Mux26:<br>00 : Select .0 input for Mux26<br>01 : Select .1 input for Mux26<br>10 : Select .2 input for Mux26<br>11 : Select .3 input for Mux26<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-92. OUTPUT2MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19-18 | MUX25 | R/W | 0h | Select Bits for OUTPUT2 Mux25:<br>00 : Select .0 input for Mux25<br>01 : Select .1 input for Mux25<br>10 : Select .2 input for Mux25<br>11 : Select .3 input for Mux25<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX24 | R/W | 0h | Select Bits for OUTPUT2 Mux24:<br>00 : Select .0 input for Mux24<br>01 : Select .1 input for Mux24<br>10 : Select .2 input for Mux24<br>11 : Select .3 input for Mux24<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX23 | R/W | 0h | Select Bits for OUTPUT2 Mux23:<br>00 : Select .0 input for Mux23<br>01 : Select .1 input for Mux23<br>10 : Select .2 input for Mux23<br>11 : Select .3 input for Mux23<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX22 | R/W | 0h | Select Bits for OUTPUT2 Mux22:<br>00 : Select .0 input for Mux22<br>01 : Select .1 input for Mux22<br>10 : Select .2 input for Mux22<br>11 : Select .3 input for Mux22<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX21 | R/W | 0h | Select Bits for OUTPUT2 Mux21:<br>00 : Select .0 input for Mux21<br>01 : Select .1 input for Mux21<br>10 : Select .2 input for Mux21<br>11 : Select .3 input for Mux21<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX20 | R/W | 0h | Select Bits for OUTPUT2 Mux20:<br>00 : Select .0 input for Mux20<br>01 : Select .1 input for Mux20<br>10 : Select .2 input for Mux20<br>11 : Select .3 input for Mux20<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7-6 | MUX19 | R/W | 0h | Select Bits for OUTPUT2 Mux19:<br>00 : Select .0 input for Mux19<br>01 : Select .1 input for Mux19<br>10 : Select .2 input for Mux19<br>11 : Select .3 input for Mux19<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX18 | R/W | 0h | Select Bits for OUTPUT2 Mux18:<br>00 : Select .0 input for Mux18<br>01 : Select .1 input for Mux18<br>10 : Select .2 input for Mux18<br>11 : Select .3 input for Mux18<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-92. OUTPUT2MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-2 | MUX17 | R/W | 0h | Select Bits for OUTPUT2 Mux17:<br>00 : Select .0 input for Mux17<br>01 : Select .1 input for Mux17<br>10 : Select .2 input for Mux17<br>11 : Select .3 input for Mux17<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX16 | R/W | 0h | Select Bits for OUTPUT2 Mux16:<br>00 : Select .0 input for Mux16<br>01 : Select .1 input for Mux16<br>10 : Select .2 input for Mux16<br>11 : Select .3 input for Mux16<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.6.5 OUTPUT3MUX0TO15CFG Register (Offset = 8h) [Reset = 00000000h]

OUTPUT3MUX0TO15CFG is shown in Figure 9-84 and described in Table 9-93.

Return to the Summary Table.

Output X-BAR Mux Configuration for Output 3

**Figure 9-84. 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 9-93. OUTPUT3MUX0TO15CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX15 | R/W | 0h | Select Bits for OUTPUT3 Mux15:<br>00 : Select .0 input for Mux15<br>01 : Select .1 input for Mux15<br>10 : Select .2 input for Mux15<br>11 : Select .3 input for Mux15<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX14 | R/W | 0h | Select Bits for OUTPUT3 Mux14:<br>00 : Select .0 input for Mux14<br>01 : Select .1 input for Mux14<br>10 : Select .2 input for Mux14<br>11 : Select .3 input for Mux14<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX13 | R/W | 0h | Select Bits for OUTPUT3 Mux13:<br>00 : Select .0 input for Mux13<br>01 : Select .1 input for Mux13<br>10 : Select .2 input for Mux13<br>11 : Select .3 input for Mux13<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX12 | R/W | 0h | Select Bits for OUTPUT3 Mux12:<br>00 : Select .0 input for Mux12<br>01 : Select .1 input for Mux12<br>10 : Select .2 input for Mux12<br>11 : Select .3 input for Mux12<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX11 | R/W | 0h | Select Bits for OUTPUT3 Mux11:<br>00 : Select .0 input for Mux11<br>01 : Select .1 input for Mux11<br>10 : Select .2 input for Mux11<br>11 : Select .3 input for Mux11<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21-20 | MUX10 | R/W | 0h | Select Bits for OUTPUT3 Mux10:<br>00 : Select .0 input for Mux10<br>01 : Select .1 input for Mux10<br>10 : Select .2 input for Mux10<br>11 : Select .3 input for Mux10<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-93. OUTPUT3MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19-18 | MUX9 | R/W | 0h | Select Bits for OUTPUT3 Mux9:<br>00 : Select .0 input for Mux9<br>01 : Select .1 input for Mux9<br>10 : Select .2 input for Mux9<br>11 : Select .3 input for Mux9<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX8 | R/W | 0h | Select Bits for OUTPUT3 Mux8:<br>00 : Select .0 input for Mux8<br>01 : Select .1 input for Mux8<br>10 : Select .2 input for Mux8<br>11 : Select .3 input for Mux8<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX7 | R/W | 0h | Select Bits for OUTPUT3 Mux7:<br>00 : Select .0 input for Mux7<br>01 : Select .1 input for Mux7<br>10 : Select .2 input for Mux7<br>11 : Select .3 input for Mux7<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX6 | R/W | 0h | Select Bits for OUTPUT3 Mux6:<br>00 : Select .0 input for Mux6<br>01 : Select .1 input for Mux6<br>10 : Select .2 input for Mux6<br>11 : Select .3 input for Mux6<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX5 | R/W | 0h | Select Bits for OUTPUT3 Mux5:<br>00 : Select .0 input for Mux5<br>01 : Select .1 input for Mux5<br>10 : Select .2 input for Mux5<br>11 : Select .3 input for Mux5<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX4 | R/W | 0h | Select Bits for OUTPUT3 Mux4:<br>00 : Select .0 input for Mux4<br>01 : Select .1 input for Mux4<br>10 : Select .2 input for Mux4<br>11 : Select .3 input for Mux4<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7-6 | MUX3 | R/W | 0h | Select Bits for OUTPUT3 Mux3:<br>00 : Select .0 input for Mux3<br>01 : Select .1 input for Mux3<br>10 : Select .2 input for Mux3<br>11 : Select .3 input for Mux3<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX2 | R/W | 0h | Select Bits for OUTPUT3 Mux2:<br>00 : Select .0 input for Mux2<br>01 : Select .1 input for Mux2<br>10 : Select .2 input for Mux2<br>11 : Select .3 input for Mux2<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-93. OUTPUT3MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-2 | MUX1 | R/W | 0h | Select Bits for OUTPUT3 Mux1:<br>00 : Select .0 input for Mux1<br>01 : Select .1 input for Mux1<br>10 : Select .2 input for Mux1<br>11 : Select .3 input for Mux1<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX0 | R/W | 0h | Select Bits for OUTPUT3 Mux0:<br>00 : Select .0 input for Mux0<br>01 : Select .1 input for Mux0<br>10 : Select .2 input for Mux0<br>11 : Select .3 input for Mux0<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.6.6 OUTPUT3MUX16TO31CFG Register (Offset = Ah) [Reset = 00000000h]

OUTPUT3MUX16TO31CFG is shown in Figure 9-85 and described in Table 9-94.

Return to the Summary Table.

Output X-BAR Mux Configuration for Output 3

**Figure 9-85. 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 9-94. OUTPUT3MUX16TO31CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX31 | R/W | 0h | Select Bits for OUTPUT3 Mux31:<br>00 : Select .0 input for Mux31<br>01 : Select .1 input for Mux31<br>10 : Select .2 input for Mux31<br>11 : Select .3 input for Mux31<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX30 | R/W | 0h | Select Bits for OUTPUT3 Mux30:<br>00 : Select .0 input for Mux30<br>01 : Select .1 input for Mux30<br>10 : Select .2 input for Mux30<br>11 : Select .3 input for Mux30<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX29 | R/W | 0h | Select Bits for OUTPUT3 Mux29:<br>00 : Select .0 input for Mux29<br>01 : Select .1 input for Mux29<br>10 : Select .2 input for Mux29<br>11 : Select .3 input for Mux29<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX28 | R/W | 0h | Select Bits for OUTPUT3 Mux28:<br>00 : Select .0 input for Mux28<br>01 : Select .1 input for Mux28<br>10 : Select .2 input for Mux28<br>11 : Select .3 input for Mux28<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX27 | R/W | 0h | Select Bits for OUTPUT3 Mux27:<br>00 : Select .0 input for Mux27<br>01 : Select .1 input for Mux27<br>10 : Select .2 input for Mux27<br>11 : Select .3 input for Mux27<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21-20 | MUX26 | R/W | 0h | Select Bits for OUTPUT3 Mux26:<br>00 : Select .0 input for Mux26<br>01 : Select .1 input for Mux26<br>10 : Select .2 input for Mux26<br>11 : Select .3 input for Mux26<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-94. OUTPUT3MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19-18 | MUX25 | R/W | 0h | Select Bits for OUTPUT3 Mux25:<br>00 : Select .0 input for Mux25<br>01 : Select .1 input for Mux25<br>10 : Select .2 input for Mux25<br>11 : Select .3 input for Mux25<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX24 | R/W | 0h | Select Bits for OUTPUT3 Mux24:<br>00 : Select .0 input for Mux24<br>01 : Select .1 input for Mux24<br>10 : Select .2 input for Mux24<br>11 : Select .3 input for Mux24<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX23 | R/W | 0h | Select Bits for OUTPUT3 Mux23:<br>00 : Select .0 input for Mux23<br>01 : Select .1 input for Mux23<br>10 : Select .2 input for Mux23<br>11 : Select .3 input for Mux23<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX22 | R/W | 0h | Select Bits for OUTPUT3 Mux22:<br>00 : Select .0 input for Mux22<br>01 : Select .1 input for Mux22<br>10 : Select .2 input for Mux22<br>11 : Select .3 input for Mux22<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX21 | R/W | 0h | Select Bits for OUTPUT3 Mux21:<br>00 : Select .0 input for Mux21<br>01 : Select .1 input for Mux21<br>10 : Select .2 input for Mux21<br>11 : Select .3 input for Mux21<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX20 | R/W | 0h | Select Bits for OUTPUT3 Mux20:<br>00 : Select .0 input for Mux20<br>01 : Select .1 input for Mux20<br>10 : Select .2 input for Mux20<br>11 : Select .3 input for Mux20<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7-6 | MUX19 | R/W | 0h | Select Bits for OUTPUT3 Mux19:<br>00 : Select .0 input for Mux19<br>01 : Select .1 input for Mux19<br>10 : Select .2 input for Mux19<br>11 : Select .3 input for Mux19<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX18 | R/W | 0h | Select Bits for OUTPUT3 Mux18:<br>00 : Select .0 input for Mux18<br>01 : Select .1 input for Mux18<br>10 : Select .2 input for Mux18<br>11 : Select .3 input for Mux18<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-94. OUTPUT3MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-2 | MUX17 | R/W | 0h | Select Bits for OUTPUT3 Mux17:<br>00 : Select .0 input for Mux17<br>01 : Select .1 input for Mux17<br>10 : Select .2 input for Mux17<br>11 : Select .3 input for Mux17<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX16 | R/W | 0h | Select Bits for OUTPUT3 Mux16:<br>00 : Select .0 input for Mux16<br>01 : Select .1 input for Mux16<br>10 : Select .2 input for Mux16<br>11 : Select .3 input for Mux16<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.6.7 OUTPUT4MUX0TO15CFG Register (Offset = Ch) [Reset = 00000000h]

OUTPUT4MUX0TO15CFG is shown in Figure 9-86 and described in Table 9-95.

Return to the Summary Table.

Output X-BAR Mux Configuration for Output 4

**Figure 9-86. 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 9-95. OUTPUT4MUX0TO15CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX15 | R/W | 0h | Select Bits for OUTPUT4 Mux15:<br>00 : Select .0 input for Mux15<br>01 : Select .1 input for Mux15<br>10 : Select .2 input for Mux15<br>11 : Select .3 input for Mux15<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX14 | R/W | 0h | Select Bits for OUTPUT4 Mux14:<br>00 : Select .0 input for Mux14<br>01 : Select .1 input for Mux14<br>10 : Select .2 input for Mux14<br>11 : Select .3 input for Mux14<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX13 | R/W | 0h | Select Bits for OUTPUT4 Mux13:<br>00 : Select .0 input for Mux13<br>01 : Select .1 input for Mux13<br>10 : Select .2 input for Mux13<br>11 : Select .3 input for Mux13<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX12 | R/W | 0h | Select Bits for OUTPUT4 Mux12:<br>00 : Select .0 input for Mux12<br>01 : Select .1 input for Mux12<br>10 : Select .2 input for Mux12<br>11 : Select .3 input for Mux12<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX11 | R/W | 0h | Select Bits for OUTPUT4 Mux11:<br>00 : Select .0 input for Mux11<br>01 : Select .1 input for Mux11<br>10 : Select .2 input for Mux11<br>11 : Select .3 input for Mux11<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21-20 | MUX10 | R/W | 0h | Select Bits for OUTPUT4 Mux10:<br>00 : Select .0 input for Mux10<br>01 : Select .1 input for Mux10<br>10 : Select .2 input for Mux10<br>11 : Select .3 input for Mux10<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-95. OUTPUT4MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19-18 | MUX9 | R/W | 0h | Select Bits for OUTPUT4 Mux9:<br>00 : Select .0 input for Mux9<br>01 : Select .1 input for Mux9<br>10 : Select .2 input for Mux9<br>11 : Select .3 input for Mux9<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX8 | R/W | 0h | Select Bits for OUTPUT4 Mux8:<br>00 : Select .0 input for Mux8<br>01 : Select .1 input for Mux8<br>10 : Select .2 input for Mux8<br>11 : Select .3 input for Mux8<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX7 | R/W | 0h | Select Bits for OUTPUT4 Mux7:<br>00 : Select .0 input for Mux7<br>01 : Select .1 input for Mux7<br>10 : Select .2 input for Mux7<br>11 : Select .3 input for Mux7<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX6 | R/W | 0h | Select Bits for OUTPUT4 Mux6:<br>00 : Select .0 input for Mux6<br>01 : Select .1 input for Mux6<br>10 : Select .2 input for Mux6<br>11 : Select .3 input for Mux6<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX5 | R/W | 0h | Select Bits for OUTPUT4 Mux5:<br>00 : Select .0 input for Mux5<br>01 : Select .1 input for Mux5<br>10 : Select .2 input for Mux5<br>11 : Select .3 input for Mux5<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX4 | R/W | 0h | Select Bits for OUTPUT4 Mux4:<br>00 : Select .0 input for Mux4<br>01 : Select .1 input for Mux4<br>10 : Select .2 input for Mux4<br>11 : Select .3 input for Mux4<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7-6 | MUX3 | R/W | 0h | Select Bits for OUTPUT4 Mux3:<br>00 : Select .0 input for Mux3<br>01 : Select .1 input for Mux3<br>10 : Select .2 input for Mux3<br>11 : Select .3 input for Mux3<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX2 | R/W | 0h | Select Bits for OUTPUT4 Mux2:<br>00 : Select .0 input for Mux2<br>01 : Select .1 input for Mux2<br>10 : Select .2 input for Mux2<br>11 : Select .3 input for Mux2<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-95. OUTPUT4MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-2 | MUX1 | R/W | 0h | Select Bits for OUTPUT4 Mux1:<br>00 : Select .0 input for Mux1<br>01 : Select .1 input for Mux1<br>10 : Select .2 input for Mux1<br>11 : Select .3 input for Mux1<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX0 | R/W | 0h | Select Bits for OUTPUT4 Mux0:<br>00 : Select .0 input for Mux0<br>01 : Select .1 input for Mux0<br>10 : Select .2 input for Mux0<br>11 : Select .3 input for Mux0<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.6.8 OUTPUT4MUX16TO31CFG Register (Offset = Eh) [Reset = 00000000h]

OUTPUT4MUX16TO31CFG is shown in Figure 9-87 and described in Table 9-96.

Return to the Summary Table.

Output X-BAR Mux Configuration for Output 4

**Figure 9-87. 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 9-96. OUTPUT4MUX16TO31CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX31 | R/W | 0h | Select Bits for OUTPUT4 Mux31:<br>00 : Select .0 input for Mux31<br>01 : Select .1 input for Mux31<br>10 : Select .2 input for Mux31<br>11 : Select .3 input for Mux31<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX30 | R/W | 0h | Select Bits for OUTPUT4 Mux30:<br>00 : Select .0 input for Mux30<br>01 : Select .1 input for Mux30<br>10 : Select .2 input for Mux30<br>11 : Select .3 input for Mux30<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX29 | R/W | 0h | Select Bits for OUTPUT4 Mux29:<br>00 : Select .0 input for Mux29<br>01 : Select .1 input for Mux29<br>10 : Select .2 input for Mux29<br>11 : Select .3 input for Mux29<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX28 | R/W | 0h | Select Bits for OUTPUT4 Mux28:<br>00 : Select .0 input for Mux28<br>01 : Select .1 input for Mux28<br>10 : Select .2 input for Mux28<br>11 : Select .3 input for Mux28<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX27 | R/W | 0h | Select Bits for OUTPUT4 Mux27:<br>00 : Select .0 input for Mux27<br>01 : Select .1 input for Mux27<br>10 : Select .2 input for Mux27<br>11 : Select .3 input for Mux27<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21-20 | MUX26 | R/W | 0h | Select Bits for OUTPUT4 Mux26:<br>00 : Select .0 input for Mux26<br>01 : Select .1 input for Mux26<br>10 : Select .2 input for Mux26<br>11 : Select .3 input for Mux26<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-96. OUTPUT4MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19-18 | MUX25 | R/W | 0h | Select Bits for OUTPUT4 Mux25:<br>00 : Select .0 input for Mux25<br>01 : Select .1 input for Mux25<br>10 : Select .2 input for Mux25<br>11 : Select .3 input for Mux25<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX24 | R/W | 0h | Select Bits for OUTPUT4 Mux24:<br>00 : Select .0 input for Mux24<br>01 : Select .1 input for Mux24<br>10 : Select .2 input for Mux24<br>11 : Select .3 input for Mux24<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX23 | R/W | 0h | Select Bits for OUTPUT4 Mux23:<br>00 : Select .0 input for Mux23<br>01 : Select .1 input for Mux23<br>10 : Select .2 input for Mux23<br>11 : Select .3 input for Mux23<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX22 | R/W | 0h | Select Bits for OUTPUT4 Mux22:<br>00 : Select .0 input for Mux22<br>01 : Select .1 input for Mux22<br>10 : Select .2 input for Mux22<br>11 : Select .3 input for Mux22<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX21 | R/W | 0h | Select Bits for OUTPUT4 Mux21:<br>00 : Select .0 input for Mux21<br>01 : Select .1 input for Mux21<br>10 : Select .2 input for Mux21<br>11 : Select .3 input for Mux21<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX20 | R/W | 0h | Select Bits for OUTPUT4 Mux20:<br>00 : Select .0 input for Mux20<br>01 : Select .1 input for Mux20<br>10 : Select .2 input for Mux20<br>11 : Select .3 input for Mux20<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7-6 | MUX19 | R/W | 0h | Select Bits for OUTPUT4 Mux19:<br>00 : Select .0 input for Mux19<br>01 : Select .1 input for Mux19<br>10 : Select .2 input for Mux19<br>11 : Select .3 input for Mux19<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX18 | R/W | 0h | Select Bits for OUTPUT4 Mux18:<br>00 : Select .0 input for Mux18<br>01 : Select .1 input for Mux18<br>10 : Select .2 input for Mux18<br>11 : Select .3 input for Mux18<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-96. OUTPUT4MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-2 | MUX17 | R/W | 0h | Select Bits for OUTPUT4 Mux17:<br>00 : Select .0 input for Mux17<br>01 : Select .1 input for Mux17<br>10 : Select .2 input for Mux17<br>11 : Select .3 input for Mux17<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX16 | R/W | 0h | Select Bits for OUTPUT4 Mux16:<br>00 : Select .0 input for Mux16<br>01 : Select .1 input for Mux16<br>10 : Select .2 input for Mux16<br>11 : Select .3 input for Mux16<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.6.9 OUTPUT5MUX0TO15CFG Register (Offset = 10h) [Reset = 00000000h]

OUTPUT5MUX0TO15CFG is shown in Figure 9-88 and described in Table 9-97.

Return to the Summary Table.

Output X-BAR Mux Configuration for Output 5

**Figure 9-88. 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 9-97. OUTPUT5MUX0TO15CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX15 | R/W | 0h | Select Bits for OUTPUT5 Mux15:<br>00 : Select .0 input for Mux15<br>01 : Select .1 input for Mux15<br>10 : Select .2 input for Mux15<br>11 : Select .3 input for Mux15<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX14 | R/W | 0h | Select Bits for OUTPUT5 Mux14:<br>00 : Select .0 input for Mux14<br>01 : Select .1 input for Mux14<br>10 : Select .2 input for Mux14<br>11 : Select .3 input for Mux14<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX13 | R/W | 0h | Select Bits for OUTPUT5 Mux13:<br>00 : Select .0 input for Mux13<br>01 : Select .1 input for Mux13<br>10 : Select .2 input for Mux13<br>11 : Select .3 input for Mux13<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX12 | R/W | 0h | Select Bits for OUTPUT5 Mux12:<br>00 : Select .0 input for Mux12<br>01 : Select .1 input for Mux12<br>10 : Select .2 input for Mux12<br>11 : Select .3 input for Mux12<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX11 | R/W | 0h | Select Bits for OUTPUT5 Mux11:<br>00 : Select .0 input for Mux11<br>01 : Select .1 input for Mux11<br>10 : Select .2 input for Mux11<br>11 : Select .3 input for Mux11<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21-20 | MUX10 | R/W | 0h | Select Bits for OUTPUT5 Mux10:<br>00 : Select .0 input for Mux10<br>01 : Select .1 input for Mux10<br>10 : Select .2 input for Mux10<br>11 : Select .3 input for Mux10<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-97. OUTPUT5MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19-18 | MUX9 | R/W | 0h | Select Bits for OUTPUT5 Mux9:<br>00 : Select .0 input for Mux9<br>01 : Select .1 input for Mux9<br>10 : Select .2 input for Mux9<br>11 : Select .3 input for Mux9<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX8 | R/W | 0h | Select Bits for OUTPUT5 Mux8:<br>00 : Select .0 input for Mux8<br>01 : Select .1 input for Mux8<br>10 : Select .2 input for Mux8<br>11 : Select .3 input for Mux8<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX7 | R/W | 0h | Select Bits for OUTPUT5 Mux7:<br>00 : Select .0 input for Mux7<br>01 : Select .1 input for Mux7<br>10 : Select .2 input for Mux7<br>11 : Select .3 input for Mux7<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX6 | R/W | 0h | Select Bits for OUTPUT5 Mux6:<br>00 : Select .0 input for Mux6<br>01 : Select .1 input for Mux6<br>10 : Select .2 input for Mux6<br>11 : Select .3 input for Mux6<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX5 | R/W | 0h | Select Bits for OUTPUT5 Mux5:<br>00 : Select .0 input for Mux5<br>01 : Select .1 input for Mux5<br>10 : Select .2 input for Mux5<br>11 : Select .3 input for Mux5<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX4 | R/W | 0h | Select Bits for OUTPUT5 Mux4:<br>00 : Select .0 input for Mux4<br>01 : Select .1 input for Mux4<br>10 : Select .2 input for Mux4<br>11 : Select .3 input for Mux4<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7-6 | MUX3 | R/W | 0h | Select Bits for OUTPUT5 Mux3:<br>00 : Select .0 input for Mux3<br>01 : Select .1 input for Mux3<br>10 : Select .2 input for Mux3<br>11 : Select .3 input for Mux3<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX2 | R/W | 0h | Select Bits for OUTPUT5 Mux2:<br>00 : Select .0 input for Mux2<br>01 : Select .1 input for Mux2<br>10 : Select .2 input for Mux2<br>11 : Select .3 input for Mux2<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-97. OUTPUT5MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-2 | MUX1 | R/W | 0h | Select Bits for OUTPUT5 Mux1:<br>00 : Select .0 input for Mux1<br>01 : Select .1 input for Mux1<br>10 : Select .2 input for Mux1<br>11 : Select .3 input for Mux1<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX0 | R/W | 0h | Select Bits for OUTPUT5 Mux0:<br>00 : Select .0 input for Mux0<br>01 : Select .1 input for Mux0<br>10 : Select .2 input for Mux0<br>11 : Select .3 input for Mux0<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.6.10 OUTPUT5MUX16TO31CFG Register (Offset = 12h) [Reset = 00000000h]

OUTPUT5MUX16TO31CFG is shown in Figure 9-89 and described in Table 9-98.

Return to the Summary Table.

Output X-BAR Mux Configuration for Output 5

**Figure 9-89. 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 9-98. OUTPUT5MUX16TO31CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX31 | R/W | 0h | Select Bits for OUTPUT5 Mux31:<br>00 : Select .0 input for Mux31<br>01 : Select .1 input for Mux31<br>10 : Select .2 input for Mux31<br>11 : Select .3 input for Mux31<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX30 | R/W | 0h | Select Bits for OUTPUT5 Mux30:<br>00 : Select .0 input for Mux30<br>01 : Select .1 input for Mux30<br>10 : Select .2 input for Mux30<br>11 : Select .3 input for Mux30<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX29 | R/W | 0h | Select Bits for OUTPUT5 Mux29:<br>00 : Select .0 input for Mux29<br>01 : Select .1 input for Mux29<br>10 : Select .2 input for Mux29<br>11 : Select .3 input for Mux29<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX28 | R/W | 0h | Select Bits for OUTPUT5 Mux28:<br>00 : Select .0 input for Mux28<br>01 : Select .1 input for Mux28<br>10 : Select .2 input for Mux28<br>11 : Select .3 input for Mux28<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX27 | R/W | 0h | Select Bits for OUTPUT5 Mux27:<br>00 : Select .0 input for Mux27<br>01 : Select .1 input for Mux27<br>10 : Select .2 input for Mux27<br>11 : Select .3 input for Mux27<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21-20 | MUX26 | R/W | 0h | Select Bits for OUTPUT5 Mux26:<br>00 : Select .0 input for Mux26<br>01 : Select .1 input for Mux26<br>10 : Select .2 input for Mux26<br>11 : Select .3 input for Mux26<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-98. OUTPUT5MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19-18 | MUX25 | R/W | 0h | Select Bits for OUTPUT5 Mux25:<br>00 : Select .0 input for Mux25<br>01 : Select .1 input for Mux25<br>10 : Select .2 input for Mux25<br>11 : Select .3 input for Mux25<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX24 | R/W | 0h | Select Bits for OUTPUT5 Mux24:<br>00 : Select .0 input for Mux24<br>01 : Select .1 input for Mux24<br>10 : Select .2 input for Mux24<br>11 : Select .3 input for Mux24<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX23 | R/W | 0h | Select Bits for OUTPUT5 Mux23:<br>00 : Select .0 input for Mux23<br>01 : Select .1 input for Mux23<br>10 : Select .2 input for Mux23<br>11 : Select .3 input for Mux23<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX22 | R/W | 0h | Select Bits for OUTPUT5 Mux22:<br>00 : Select .0 input for Mux22<br>01 : Select .1 input for Mux22<br>10 : Select .2 input for Mux22<br>11 : Select .3 input for Mux22<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX21 | R/W | 0h | Select Bits for OUTPUT5 Mux21:<br>00 : Select .0 input for Mux21<br>01 : Select .1 input for Mux21<br>10 : Select .2 input for Mux21<br>11 : Select .3 input for Mux21<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX20 | R/W | 0h | Select Bits for OUTPUT5 Mux20:<br>00 : Select .0 input for Mux20<br>01 : Select .1 input for Mux20<br>10 : Select .2 input for Mux20<br>11 : Select .3 input for Mux20<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7-6 | MUX19 | R/W | 0h | Select Bits for OUTPUT5 Mux19:<br>00 : Select .0 input for Mux19<br>01 : Select .1 input for Mux19<br>10 : Select .2 input for Mux19<br>11 : Select .3 input for Mux19<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX18 | R/W | 0h | Select Bits for OUTPUT5 Mux18:<br>00 : Select .0 input for Mux18<br>01 : Select .1 input for Mux18<br>10 : Select .2 input for Mux18<br>11 : Select .3 input for Mux18<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-98. OUTPUT5MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-2 | MUX17 | R/W | 0h | Select Bits for OUTPUT5 Mux17:<br>00 : Select .0 input for Mux17<br>01 : Select .1 input for Mux17<br>10 : Select .2 input for Mux17<br>11 : Select .3 input for Mux17<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX16 | R/W | 0h | Select Bits for OUTPUT5 Mux16:<br>00 : Select .0 input for Mux16<br>01 : Select .1 input for Mux16<br>10 : Select .2 input for Mux16<br>11 : Select .3 input for Mux16<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.6.11 OUTPUT6MUX0TO15CFG Register (Offset = 14h) [Reset = 00000000h]

OUTPUT6MUX0TO15CFG is shown in Figure 9-90 and described in Table 9-99.

Return to the Summary Table.

Output X-BAR Mux Configuration for Output 6

**Figure 9-90. 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 9-99. OUTPUT6MUX0TO15CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX15 | R/W | 0h | Select Bits for OUTPUT6 Mux15:<br>00 : Select .0 input for Mux15<br>01 : Select .1 input for Mux15<br>10 : Select .2 input for Mux15<br>11 : Select .3 input for Mux15<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX14 | R/W | 0h | Select Bits for OUTPUT6 Mux14:<br>00 : Select .0 input for Mux14<br>01 : Select .1 input for Mux14<br>10 : Select .2 input for Mux14<br>11 : Select .3 input for Mux14<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX13 | R/W | 0h | Select Bits for OUTPUT6 Mux13:<br>00 : Select .0 input for Mux13<br>01 : Select .1 input for Mux13<br>10 : Select .2 input for Mux13<br>11 : Select .3 input for Mux13<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX12 | R/W | 0h | Select Bits for OUTPUT6 Mux12:<br>00 : Select .0 input for Mux12<br>01 : Select .1 input for Mux12<br>10 : Select .2 input for Mux12<br>11 : Select .3 input for Mux12<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX11 | R/W | 0h | Select Bits for OUTPUT6 Mux11:<br>00 : Select .0 input for Mux11<br>01 : Select .1 input for Mux11<br>10 : Select .2 input for Mux11<br>11 : Select .3 input for Mux11<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21-20 | MUX10 | R/W | 0h | Select Bits for OUTPUT6 Mux10:<br>00 : Select .0 input for Mux10<br>01 : Select .1 input for Mux10<br>10 : Select .2 input for Mux10<br>11 : Select .3 input for Mux10<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-99. OUTPUT6MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19-18 | MUX9 | R/W | 0h | Select Bits for OUTPUT6 Mux9:<br>00 : Select .0 input for Mux9<br>01 : Select .1 input for Mux9<br>10 : Select .2 input for Mux9<br>11 : Select .3 input for Mux9<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX8 | R/W | 0h | Select Bits for OUTPUT6 Mux8:<br>00 : Select .0 input for Mux8<br>01 : Select .1 input for Mux8<br>10 : Select .2 input for Mux8<br>11 : Select .3 input for Mux8<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX7 | R/W | 0h | Select Bits for OUTPUT6 Mux7:<br>00 : Select .0 input for Mux7<br>01 : Select .1 input for Mux7<br>10 : Select .2 input for Mux7<br>11 : Select .3 input for Mux7<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX6 | R/W | 0h | Select Bits for OUTPUT6 Mux6:<br>00 : Select .0 input for Mux6<br>01 : Select .1 input for Mux6<br>10 : Select .2 input for Mux6<br>11 : Select .3 input for Mux6<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX5 | R/W | 0h | Select Bits for OUTPUT6 Mux5:<br>00 : Select .0 input for Mux5<br>01 : Select .1 input for Mux5<br>10 : Select .2 input for Mux5<br>11 : Select .3 input for Mux5<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX4 | R/W | 0h | Select Bits for OUTPUT6 Mux4:<br>00 : Select .0 input for Mux4<br>01 : Select .1 input for Mux4<br>10 : Select .2 input for Mux4<br>11 : Select .3 input for Mux4<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7-6 | MUX3 | R/W | 0h | Select Bits for OUTPUT6 Mux3:<br>00 : Select .0 input for Mux3<br>01 : Select .1 input for Mux3<br>10 : Select .2 input for Mux3<br>11 : Select .3 input for Mux3<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX2 | R/W | 0h | Select Bits for OUTPUT6 Mux2:<br>00 : Select .0 input for Mux2<br>01 : Select .1 input for Mux2<br>10 : Select .2 input for Mux2<br>11 : Select .3 input for Mux2<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-99. OUTPUT6MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-2 | MUX1 | R/W | 0h | Select Bits for OUTPUT6 Mux1:<br>00 : Select .0 input for Mux1<br>01 : Select .1 input for Mux1<br>10 : Select .2 input for Mux1<br>11 : Select .3 input for Mux1<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX0 | R/W | 0h | Select Bits for OUTPUT6 Mux0:<br>00 : Select .0 input for Mux0<br>01 : Select .1 input for Mux0<br>10 : Select .2 input for Mux0<br>11 : Select .3 input for Mux0<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.6.12 OUTPUT6MUX16TO31CFG Register (Offset = 16h) [Reset = 00000000h]

OUTPUT6MUX16TO31CFG is shown in Figure 9-91 and described in Table 9-100.

Return to the Summary Table.

Output X-BAR Mux Configuration for Output 6

**Figure 9-91. 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 9-100. OUTPUT6MUX16TO31CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX31 | R/W | 0h | Select Bits for OUTPUT6 Mux31:<br>00 : Select .0 input for Mux31<br>01 : Select .1 input for Mux31<br>10 : Select .2 input for Mux31<br>11 : Select .3 input for Mux31<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX30 | R/W | 0h | Select Bits for OUTPUT6 Mux30:<br>00 : Select .0 input for Mux30<br>01 : Select .1 input for Mux30<br>10 : Select .2 input for Mux30<br>11 : Select .3 input for Mux30<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX29 | R/W | 0h | Select Bits for OUTPUT6 Mux29:<br>00 : Select .0 input for Mux29<br>01 : Select .1 input for Mux29<br>10 : Select .2 input for Mux29<br>11 : Select .3 input for Mux29<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX28 | R/W | 0h | Select Bits for OUTPUT6 Mux28:<br>00 : Select .0 input for Mux28<br>01 : Select .1 input for Mux28<br>10 : Select .2 input for Mux28<br>11 : Select .3 input for Mux28<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX27 | R/W | 0h | Select Bits for OUTPUT6 Mux27:<br>00 : Select .0 input for Mux27<br>01 : Select .1 input for Mux27<br>10 : Select .2 input for Mux27<br>11 : Select .3 input for Mux27<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21-20 | MUX26 | R/W | 0h | Select Bits for OUTPUT6 Mux26:<br>00 : Select .0 input for Mux26<br>01 : Select .1 input for Mux26<br>10 : Select .2 input for Mux26<br>11 : Select .3 input for Mux26<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-100. OUTPUT6MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19-18 | MUX25 | R/W | 0h | Select Bits for OUTPUT6 Mux25:<br>00 : Select .0 input for Mux25<br>01 : Select .1 input for Mux25<br>10 : Select .2 input for Mux25<br>11 : Select .3 input for Mux25<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX24 | R/W | 0h | Select Bits for OUTPUT6 Mux24:<br>00 : Select .0 input for Mux24<br>01 : Select .1 input for Mux24<br>10 : Select .2 input for Mux24<br>11 : Select .3 input for Mux24<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX23 | R/W | 0h | Select Bits for OUTPUT6 Mux23:<br>00 : Select .0 input for Mux23<br>01 : Select .1 input for Mux23<br>10 : Select .2 input for Mux23<br>11 : Select .3 input for Mux23<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX22 | R/W | 0h | Select Bits for OUTPUT6 Mux22:<br>00 : Select .0 input for Mux22<br>01 : Select .1 input for Mux22<br>10 : Select .2 input for Mux22<br>11 : Select .3 input for Mux22<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX21 | R/W | 0h | Select Bits for OUTPUT6 Mux21:<br>00 : Select .0 input for Mux21<br>01 : Select .1 input for Mux21<br>10 : Select .2 input for Mux21<br>11 : Select .3 input for Mux21<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX20 | R/W | 0h | Select Bits for OUTPUT6 Mux20:<br>00 : Select .0 input for Mux20<br>01 : Select .1 input for Mux20<br>10 : Select .2 input for Mux20<br>11 : Select .3 input for Mux20<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7-6 | MUX19 | R/W | 0h | Select Bits for OUTPUT6 Mux19:<br>00 : Select .0 input for Mux19<br>01 : Select .1 input for Mux19<br>10 : Select .2 input for Mux19<br>11 : Select .3 input for Mux19<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX18 | R/W | 0h | Select Bits for OUTPUT6 Mux18:<br>00 : Select .0 input for Mux18<br>01 : Select .1 input for Mux18<br>10 : Select .2 input for Mux18<br>11 : Select .3 input for Mux18<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-100. OUTPUT6MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-2 | MUX17 | R/W | 0h | Select Bits for OUTPUT6 Mux17:<br>00 : Select .0 input for Mux17<br>01 : Select .1 input for Mux17<br>10 : Select .2 input for Mux17<br>11 : Select .3 input for Mux17<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX16 | R/W | 0h | Select Bits for OUTPUT6 Mux16:<br>00 : Select .0 input for Mux16<br>01 : Select .1 input for Mux16<br>10 : Select .2 input for Mux16<br>11 : Select .3 input for Mux16<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.6.13 OUTPUT7MUX0TO15CFG Register (Offset = 18h) [Reset = 00000000h]

OUTPUT7MUX0TO15CFG is shown in Figure 9-92 and described in Table 9-101.

Return to the Summary Table.

Output X-BAR Mux Configuration for Output 7

**Figure 9-92. 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 9-101. OUTPUT7MUX0TO15CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX15 | R/W | 0h | Select Bits for OUTPUT7 Mux15:<br>00 : Select .0 input for Mux15<br>01 : Select .1 input for Mux15<br>10 : Select .2 input for Mux15<br>11 : Select .3 input for Mux15<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX14 | R/W | 0h | Select Bits for OUTPUT7 Mux14:<br>00 : Select .0 input for Mux14<br>01 : Select .1 input for Mux14<br>10 : Select .2 input for Mux14<br>11 : Select .3 input for Mux14<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX13 | R/W | 0h | Select Bits for OUTPUT7 Mux13:<br>00 : Select .0 input for Mux13<br>01 : Select .1 input for Mux13<br>10 : Select .2 input for Mux13<br>11 : Select .3 input for Mux13<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX12 | R/W | 0h | Select Bits for OUTPUT7 Mux12:<br>00 : Select .0 input for Mux12<br>01 : Select .1 input for Mux12<br>10 : Select .2 input for Mux12<br>11 : Select .3 input for Mux12<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX11 | R/W | 0h | Select Bits for OUTPUT7 Mux11:<br>00 : Select .0 input for Mux11<br>01 : Select .1 input for Mux11<br>10 : Select .2 input for Mux11<br>11 : Select .3 input for Mux11<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21-20 | MUX10 | R/W | 0h | Select Bits for OUTPUT7 Mux10:<br>00 : Select .0 input for Mux10<br>01 : Select .1 input for Mux10<br>10 : Select .2 input for Mux10<br>11 : Select .3 input for Mux10<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-101. OUTPUT7MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19-18 | MUX9 | R/W | 0h | Select Bits for OUTPUT7 Mux9:<br>00 : Select .0 input for Mux9<br>01 : Select .1 input for Mux9<br>10 : Select .2 input for Mux9<br>11 : Select .3 input for Mux9<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX8 | R/W | 0h | Select Bits for OUTPUT7 Mux8:<br>00 : Select .0 input for Mux8<br>01 : Select .1 input for Mux8<br>10 : Select .2 input for Mux8<br>11 : Select .3 input for Mux8<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX7 | R/W | 0h | Select Bits for OUTPUT7 Mux7:<br>00 : Select .0 input for Mux7<br>01 : Select .1 input for Mux7<br>10 : Select .2 input for Mux7<br>11 : Select .3 input for Mux7<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX6 | R/W | 0h | Select Bits for OUTPUT7 Mux6:<br>00 : Select .0 input for Mux6<br>01 : Select .1 input for Mux6<br>10 : Select .2 input for Mux6<br>11 : Select .3 input for Mux6<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX5 | R/W | 0h | Select Bits for OUTPUT7 Mux5:<br>00 : Select .0 input for Mux5<br>01 : Select .1 input for Mux5<br>10 : Select .2 input for Mux5<br>11 : Select .3 input for Mux5<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX4 | R/W | 0h | Select Bits for OUTPUT7 Mux4:<br>00 : Select .0 input for Mux4<br>01 : Select .1 input for Mux4<br>10 : Select .2 input for Mux4<br>11 : Select .3 input for Mux4<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7-6 | MUX3 | R/W | 0h | Select Bits for OUTPUT7 Mux3:<br>00 : Select .0 input for Mux3<br>01 : Select .1 input for Mux3<br>10 : Select .2 input for Mux3<br>11 : Select .3 input for Mux3<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX2 | R/W | 0h | Select Bits for OUTPUT7 Mux2:<br>00 : Select .0 input for Mux2<br>01 : Select .1 input for Mux2<br>10 : Select .2 input for Mux2<br>11 : Select .3 input for Mux2<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-101. OUTPUT7MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-2 | MUX1 | R/W | 0h | Select Bits for OUTPUT7 Mux1:<br>00 : Select .0 input for Mux1<br>01 : Select .1 input for Mux1<br>10 : Select .2 input for Mux1<br>11 : Select .3 input for Mux1<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX0 | R/W | 0h | Select Bits for OUTPUT7 Mux0:<br>00 : Select .0 input for Mux0<br>01 : Select .1 input for Mux0<br>10 : Select .2 input for Mux0<br>11 : Select .3 input for Mux0<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.6.14 OUTPUT7MUX16TO31CFG Register (Offset = 1Ah) [Reset = 00000000h]

OUTPUT7MUX16TO31CFG is shown in Figure 9-93 and described in Table 9-102.

Return to the Summary Table.

Output X-BAR Mux Configuration for Output 7

**Figure 9-93. 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 9-102. OUTPUT7MUX16TO31CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX31 | R/W | 0h | Select Bits for OUTPUT7 Mux31:<br>00 : Select .0 input for Mux31<br>01 : Select .1 input for Mux31<br>10 : Select .2 input for Mux31<br>11 : Select .3 input for Mux31<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX30 | R/W | 0h | Select Bits for OUTPUT7 Mux30:<br>00 : Select .0 input for Mux30<br>01 : Select .1 input for Mux30<br>10 : Select .2 input for Mux30<br>11 : Select .3 input for Mux30<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX29 | R/W | 0h | Select Bits for OUTPUT7 Mux29:<br>00 : Select .0 input for Mux29<br>01 : Select .1 input for Mux29<br>10 : Select .2 input for Mux29<br>11 : Select .3 input for Mux29<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX28 | R/W | 0h | Select Bits for OUTPUT7 Mux28:<br>00 : Select .0 input for Mux28<br>01 : Select .1 input for Mux28<br>10 : Select .2 input for Mux28<br>11 : Select .3 input for Mux28<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX27 | R/W | 0h | Select Bits for OUTPUT7 Mux27:<br>00 : Select .0 input for Mux27<br>01 : Select .1 input for Mux27<br>10 : Select .2 input for Mux27<br>11 : Select .3 input for Mux27<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21-20 | MUX26 | R/W | 0h | Select Bits for OUTPUT7 Mux26:<br>00 : Select .0 input for Mux26<br>01 : Select .1 input for Mux26<br>10 : Select .2 input for Mux26<br>11 : Select .3 input for Mux26<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-102. OUTPUT7MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19-18 | MUX25 | R/W | 0h | Select Bits for OUTPUT7 Mux25:<br>00 : Select .0 input for Mux25<br>01 : Select .1 input for Mux25<br>10 : Select .2 input for Mux25<br>11 : Select .3 input for Mux25<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX24 | R/W | 0h | Select Bits for OUTPUT7 Mux24:<br>00 : Select .0 input for Mux24<br>01 : Select .1 input for Mux24<br>10 : Select .2 input for Mux24<br>11 : Select .3 input for Mux24<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX23 | R/W | 0h | Select Bits for OUTPUT7 Mux23:<br>00 : Select .0 input for Mux23<br>01 : Select .1 input for Mux23<br>10 : Select .2 input for Mux23<br>11 : Select .3 input for Mux23<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX22 | R/W | 0h | Select Bits for OUTPUT7 Mux22:<br>00 : Select .0 input for Mux22<br>01 : Select .1 input for Mux22<br>10 : Select .2 input for Mux22<br>11 : Select .3 input for Mux22<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX21 | R/W | 0h | Select Bits for OUTPUT7 Mux21:<br>00 : Select .0 input for Mux21<br>01 : Select .1 input for Mux21<br>10 : Select .2 input for Mux21<br>11 : Select .3 input for Mux21<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX20 | R/W | 0h | Select Bits for OUTPUT7 Mux20:<br>00 : Select .0 input for Mux20<br>01 : Select .1 input for Mux20<br>10 : Select .2 input for Mux20<br>11 : Select .3 input for Mux20<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7-6 | MUX19 | R/W | 0h | Select Bits for OUTPUT7 Mux19:<br>00 : Select .0 input for Mux19<br>01 : Select .1 input for Mux19<br>10 : Select .2 input for Mux19<br>11 : Select .3 input for Mux19<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX18 | R/W | 0h | Select Bits for OUTPUT7 Mux18:<br>00 : Select .0 input for Mux18<br>01 : Select .1 input for Mux18<br>10 : Select .2 input for Mux18<br>11 : Select .3 input for Mux18<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-102. OUTPUT7MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-2 | MUX17 | R/W | 0h | Select Bits for OUTPUT7 Mux17:<br>00 : Select .0 input for Mux17<br>01 : Select .1 input for Mux17<br>10 : Select .2 input for Mux17<br>11 : Select .3 input for Mux17<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX16 | R/W | 0h | Select Bits for OUTPUT7 Mux16:<br>00 : Select .0 input for Mux16<br>01 : Select .1 input for Mux16<br>10 : Select .2 input for Mux16<br>11 : Select .3 input for Mux16<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.6.15 OUTPUT8MUX0TO15CFG Register (Offset = 1Ch) [Reset = 00000000h]

OUTPUT8MUX0TO15CFG is shown in Figure 9-94 and described in Table 9-103.

Return to the Summary Table.

Output X-BAR Mux Configuration for Output 8

**Figure 9-94. 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 9-103. OUTPUT8MUX0TO15CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX15 | R/W | 0h | Select Bits for OUTPUT8 Mux15:<br>00 : Select .0 input for Mux15<br>01 : Select .1 input for Mux15<br>10 : Select .2 input for Mux15<br>11 : Select .3 input for Mux15<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX14 | R/W | 0h | Select Bits for OUTPUT8 Mux14:<br>00 : Select .0 input for Mux14<br>01 : Select .1 input for Mux14<br>10 : Select .2 input for Mux14<br>11 : Select .3 input for Mux14<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX13 | R/W | 0h | Select Bits for OUTPUT8 Mux13:<br>00 : Select .0 input for Mux13<br>01 : Select .1 input for Mux13<br>10 : Select .2 input for Mux13<br>11 : Select .3 input for Mux13<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX12 | R/W | 0h | Select Bits for OUTPUT8 Mux12:<br>00 : Select .0 input for Mux12<br>01 : Select .1 input for Mux12<br>10 : Select .2 input for Mux12<br>11 : Select .3 input for Mux12<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX11 | R/W | 0h | Select Bits for OUTPUT8 Mux11:<br>00 : Select .0 input for Mux11<br>01 : Select .1 input for Mux11<br>10 : Select .2 input for Mux11<br>11 : Select .3 input for Mux11<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21-20 | MUX10 | R/W | 0h | Select Bits for OUTPUT8 Mux10:<br>00 : Select .0 input for Mux10<br>01 : Select .1 input for Mux10<br>10 : Select .2 input for Mux10<br>11 : Select .3 input for Mux10<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-103. OUTPUT8MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19-18 | MUX9 | R/W | 0h | Select Bits for OUTPUT8 Mux9:<br>00 : Select .0 input for Mux9<br>01 : Select .1 input for Mux9<br>10 : Select .2 input for Mux9<br>11 : Select .3 input for Mux9<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX8 | R/W | 0h | Select Bits for OUTPUT8 Mux8:<br>00 : Select .0 input for Mux8<br>01 : Select .1 input for Mux8<br>10 : Select .2 input for Mux8<br>11 : Select .3 input for Mux8<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX7 | R/W | 0h | Select Bits for OUTPUT8 Mux7:<br>00 : Select .0 input for Mux7<br>01 : Select .1 input for Mux7<br>10 : Select .2 input for Mux7<br>11 : Select .3 input for Mux7<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX6 | R/W | 0h | Select Bits for OUTPUT8 Mux6:<br>00 : Select .0 input for Mux6<br>01 : Select .1 input for Mux6<br>10 : Select .2 input for Mux6<br>11 : Select .3 input for Mux6<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX5 | R/W | 0h | Select Bits for OUTPUT8 Mux5:<br>00 : Select .0 input for Mux5<br>01 : Select .1 input for Mux5<br>10 : Select .2 input for Mux5<br>11 : Select .3 input for Mux5<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX4 | R/W | 0h | Select Bits for OUTPUT8 Mux4:<br>00 : Select .0 input for Mux4<br>01 : Select .1 input for Mux4<br>10 : Select .2 input for Mux4<br>11 : Select .3 input for Mux4<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7-6 | MUX3 | R/W | 0h | Select Bits for OUTPUT8 Mux3:<br>00 : Select .0 input for Mux3<br>01 : Select .1 input for Mux3<br>10 : Select .2 input for Mux3<br>11 : Select .3 input for Mux3<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX2 | R/W | 0h | Select Bits for OUTPUT8 Mux2:<br>00 : Select .0 input for Mux2<br>01 : Select .1 input for Mux2<br>10 : Select .2 input for Mux2<br>11 : Select .3 input for Mux2<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-103. OUTPUT8MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-2 | MUX1 | R/W | 0h | Select Bits for OUTPUT8 Mux1:<br>00 : Select .0 input for Mux1<br>01 : Select .1 input for Mux1<br>10 : Select .2 input for Mux1<br>11 : Select .3 input for Mux1<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX0 | R/W | 0h | Select Bits for OUTPUT8 Mux0:<br>00 : Select .0 input for Mux0<br>01 : Select .1 input for Mux0<br>10 : Select .2 input for Mux0<br>11 : Select .3 input for Mux0<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.6.16 OUTPUT8MUX16TO31CFG Register (Offset = 1Eh) [Reset = 00000000h]

OUTPUT8MUX16TO31CFG is shown in Figure 9-95 and described in Table 9-104.

Return to the Summary Table.

Output X-BAR Mux Configuration for Output 8

**Figure 9-95. 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 9-104. OUTPUT8MUX16TO31CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX31 | R/W | 0h | Select Bits for OUTPUT8 Mux31:<br>00 : Select .0 input for Mux31<br>01 : Select .1 input for Mux31<br>10 : Select .2 input for Mux31<br>11 : Select .3 input for Mux31<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29-28 | MUX30 | R/W | 0h | Select Bits for OUTPUT8 Mux30:<br>00 : Select .0 input for Mux30<br>01 : Select .1 input for Mux30<br>10 : Select .2 input for Mux30<br>11 : Select .3 input for Mux30<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 27-26 | MUX29 | R/W | 0h | Select Bits for OUTPUT8 Mux29:<br>00 : Select .0 input for Mux29<br>01 : Select .1 input for Mux29<br>10 : Select .2 input for Mux29<br>11 : Select .3 input for Mux29<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25-24 | MUX28 | R/W | 0h | Select Bits for OUTPUT8 Mux28:<br>00 : Select .0 input for Mux28<br>01 : Select .1 input for Mux28<br>10 : Select .2 input for Mux28<br>11 : Select .3 input for Mux28<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23-22 | MUX27 | R/W | 0h | Select Bits for OUTPUT8 Mux27:<br>00 : Select .0 input for Mux27<br>01 : Select .1 input for Mux27<br>10 : Select .2 input for Mux27<br>11 : Select .3 input for Mux27<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21-20 | MUX26 | R/W | 0h | Select Bits for OUTPUT8 Mux26:<br>00 : Select .0 input for Mux26<br>01 : Select .1 input for Mux26<br>10 : Select .2 input for Mux26<br>11 : Select .3 input for Mux26<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-104. OUTPUT8MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19-18 | MUX25 | R/W | 0h | Select Bits for OUTPUT8 Mux25:<br>00 : Select .0 input for Mux25<br>01 : Select .1 input for Mux25<br>10 : Select .2 input for Mux25<br>11 : Select .3 input for Mux25<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17-16 | MUX24 | R/W | 0h | Select Bits for OUTPUT8 Mux24:<br>00 : Select .0 input for Mux24<br>01 : Select .1 input for Mux24<br>10 : Select .2 input for Mux24<br>11 : Select .3 input for Mux24<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15-14 | MUX23 | R/W | 0h | Select Bits for OUTPUT8 Mux23:<br>00 : Select .0 input for Mux23<br>01 : Select .1 input for Mux23<br>10 : Select .2 input for Mux23<br>11 : Select .3 input for Mux23<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13-12 | MUX22 | R/W | 0h | Select Bits for OUTPUT8 Mux22:<br>00 : Select .0 input for Mux22<br>01 : Select .1 input for Mux22<br>10 : Select .2 input for Mux22<br>11 : Select .3 input for Mux22<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 11-10 | MUX21 | R/W | 0h | Select Bits for OUTPUT8 Mux21:<br>00 : Select .0 input for Mux21<br>01 : Select .1 input for Mux21<br>10 : Select .2 input for Mux21<br>11 : Select .3 input for Mux21<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9-8 | MUX20 | R/W | 0h | Select Bits for OUTPUT8 Mux20:<br>00 : Select .0 input for Mux20<br>01 : Select .1 input for Mux20<br>10 : Select .2 input for Mux20<br>11 : Select .3 input for Mux20<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7-6 | MUX19 | R/W | 0h | Select Bits for OUTPUT8 Mux19:<br>00 : Select .0 input for Mux19<br>01 : Select .1 input for Mux19<br>10 : Select .2 input for Mux19<br>11 : Select .3 input for Mux19<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5-4 | MUX18 | R/W | 0h | Select Bits for OUTPUT8 Mux18:<br>00 : Select .0 input for Mux18<br>01 : Select .1 input for Mux18<br>10 : Select .2 input for Mux18<br>11 : Select .3 input for Mux18<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-104. OUTPUT8MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-2 | MUX17 | R/W | 0h | Select Bits for OUTPUT8 Mux17:<br>00 : Select .0 input for Mux17<br>01 : Select .1 input for Mux17<br>10 : Select .2 input for Mux17<br>11 : Select .3 input for Mux17<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1-0 | MUX16 | R/W | 0h | Select Bits for OUTPUT8 Mux16:<br>00 : Select .0 input for Mux16<br>01 : Select .1 input for Mux16<br>10 : Select .2 input for Mux16<br>11 : Select .3 input for Mux16<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.6.17 OUTPUT1MUXENABLE Register (Offset = 20h) [Reset = 00000000h]

OUTPUT1MUXENABLE is shown in Figure 9-96 and described in Table 9-105.

Return to the Summary Table.

Output X-BAR Mux Enable for Output 1

**Figure 9-96. OUTPUT1MUXENABLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MUX31 | MUX30 | MUX29 | MUX28 | MUX27 | MUX26 | MUX25 | MUX24 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MUX23 | MUX22 | MUX21 | MUX20 | MUX19 | MUX18 | MUX17 | MUX16 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MUX15 | MUX14 | MUX13 | MUX12 | MUX11 | MUX10 | MUX9 | MUX8 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | MUX6 | MUX5 | MUX4 | MUX3 | MUX2 | MUX1 | MUX0 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 9-105. 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<br>0: Respective output of Mux31 is disabled to drive the OUTPUT1 of OUTPUT-XBAR<br>1: Respective output of Mux31 is enabled to drive the OUTPUT1 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 30 | MUX30 | R/W | 0h | Selects the output of Mux30 to drive OUTPUT1 of OUTPUT-XBAR<br>0: Respective output of Mux30 is disabled to drive the OUTPUT1 of OUTPUT-XBAR<br>1: Respective output of Mux30 is enabled to drive the OUTPUT1 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29 | MUX29 | R/W | 0h | Selects the output of Mux29 to drive OUTPUT1 of OUTPUT-XBAR<br>0: Respective output of Mux29 is disabled to drive the OUTPUT1 of OUTPUT-XBAR<br>1: Respective output of Mux29 is enabled to drive the OUTPUT1 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 28 | MUX28 | R/W | 0h | Selects the output of Mux28 to drive OUTPUT1 of OUTPUT-XBAR<br>0: Respective output of Mux28 is disabled to drive the OUTPUT1 of OUTPUT-XBAR<br>1: Respective output of Mux28 is enabled to drive the OUTPUT1 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-105. 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<br>0: Respective output of Mux27 is disabled to drive the OUTPUT1 of OUTPUT-XBAR<br>1: Respective output of Mux27 is enabled to drive the OUTPUT1 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 26 | MUX26 | R/W | 0h | Selects the output of Mux26 to drive OUTPUT1 of OUTPUT-XBAR<br>0: Respective output of Mux26 is disabled to drive the OUTPUT1 of OUTPUT-XBAR<br>1: Respective output of Mux26 is enabled to drive the OUTPUT1 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25 | MUX25 | R/W | 0h | Selects the output of Mux25 to drive OUTPUT1 of OUTPUT-XBAR<br>0: Respective output of Mux25 is disabled to drive the OUTPUT1 of OUTPUT-XBAR<br>1: Respective output of Mux25 is enabled to drive the OUTPUT1 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 24 | MUX24 | R/W | 0h | Selects the output of Mux24 to drive OUTPUT1 of OUTPUT-XBAR<br>0: Respective output of Mux24 is disabled to drive the OUTPUT1 of OUTPUT-XBAR<br>1: Respective output of Mux24 is enabled to drive the OUTPUT1 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23 | MUX23 | R/W | 0h | Selects the output of Mux23 to drive OUTPUT1 of OUTPUT-XBAR<br>0: Respective output of Mux23 is disabled to drive the OUTPUT1 of OUTPUT-XBAR<br>1: Respective output of Mux23 is enabled to drive the OUTPUT1 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 22 | MUX22 | R/W | 0h | Selects the output of Mux22 to drive OUTPUT1 of OUTPUT-XBAR<br>0: Respective output of Mux22 is disabled to drive the OUTPUT1 of OUTPUT-XBAR<br>1: Respective output of Mux22 is enabled to drive the OUTPUT1 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21 | MUX21 | R/W | 0h | Selects the output of Mux21 to drive OUTPUT1 of OUTPUT-XBAR<br>0: Respective output of Mux21 is disabled to drive the OUTPUT1 of OUTPUT-XBAR<br>1: Respective output of Mux21 is enabled to drive the OUTPUT1 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 20 | MUX20 | R/W | 0h | Selects the output of Mux20 to drive OUTPUT1 of OUTPUT-XBAR<br>0: Respective output of Mux20 is disabled to drive the OUTPUT1 of OUTPUT-XBAR<br>1: Respective output of Mux20 is enabled to drive the OUTPUT1 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-105. 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<br>0: Respective output of Mux19 is disabled to drive the OUTPUT1 of OUTPUT-XBAR<br>1: Respective output of Mux19 is enabled to drive the OUTPUT1 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 18 | MUX18 | R/W | 0h | Selects the output of Mux18 to drive OUTPUT1 of OUTPUT-XBAR<br>0: Respective output of Mux18 is disabled to drive the OUTPUT1 of OUTPUT-XBAR<br>1: Respective output of Mux18 is enabled to drive the OUTPUT1 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17 | MUX17 | R/W | 0h | Selects the output of Mux17 to drive OUTPUT1 of OUTPUT-XBAR<br>0: Respective output of Mux17 is disabled to drive the OUTPUT1 of OUTPUT-XBAR<br>1: Respective output of Mux17 is enabled to drive the OUTPUT1 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 16 | MUX16 | R/W | 0h | Selects the output of Mux16 to drive OUTPUT1 of OUTPUT-XBAR<br>0: Respective output of Mux16 is disabled to drive the OUTPUT1 of OUTPUT-XBAR<br>1: Respective output of Mux16 is enabled to drive the OUTPUT1 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15 | MUX15 | R/W | 0h | Selects the output of Mux15 to drive OUTPUT1 of OUTPUT-XBAR<br>0: Respective output of Mux15 is disabled to drive the OUTPUT1 of OUTPUT-XBAR<br>1: Respective output of Mux15 is enabled to drive the OUTPUT1 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 14 | MUX14 | R/W | 0h | Selects the output of Mux14 to drive OUTPUT1 of OUTPUT-XBAR<br>0: Respective output of Mux14 is disabled to drive the OUTPUT1 of OUTPUT-XBAR<br>1: Respective output of Mux14 is enabled to drive the OUTPUT1 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13 | MUX13 | R/W | 0h | Selects the output of Mux13 to drive OUTPUT1 of OUTPUT-XBAR<br>0: Respective output of Mux13 is disabled to drive the OUTPUT1 of OUTPUT-XBAR<br>1: Respective output of Mux13 is enabled to drive the OUTPUT1 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 12 | MUX12 | R/W | 0h | Selects the output of Mux12 to drive OUTPUT1 of OUTPUT-XBAR<br>0: Respective output of Mux12 is disabled to drive the OUTPUT1 of OUTPUT-XBAR<br>1: Respective output of Mux12 is enabled to drive the OUTPUT1 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-105. 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<br>0: Respective output of Mux11 is disabled to drive the OUTPUT1 of OUTPUT-XBAR<br>1: Respective output of Mux11 is enabled to drive the OUTPUT1 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 10 | MUX10 | R/W | 0h | Selects the output of Mux10 to drive OUTPUT1 of OUTPUT-XBAR<br>0: Respective output of Mux10 is disabled to drive the OUTPUT1 of OUTPUT-XBAR<br>1: Respective output of Mux10 is enabled to drive the OUTPUT1 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9 | MUX9 | R/W | 0h | Selects the output of Mux9 to drive OUTPUT1 of OUTPUT-XBAR<br>0: Respective output of Mux9 is disabled to drive the OUTPUT1 of OUTPUT-XBAR<br>1: Respective output of Mux9 is enabled to drive the OUTPUT1 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 8 | MUX8 | R/W | 0h | Selects the output of Mux8 to drive OUTPUT1 of OUTPUT-XBAR<br>0: Respective output of Mux8 is disabled to drive the OUTPUT1 of OUTPUT-XBAR<br>1: Respective output of Mux8 is enabled to drive the OUTPUT1 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7 | MUX7 | R/W | 0h | Selects the output of Mux7 to drive OUTPUT1 of OUTPUT-XBAR<br>0: Respective output of Mux7 is disabled to drive the OUTPUT1 of OUTPUT-XBAR<br>1: Respective output of Mux7 is enabled to drive the OUTPUT1 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 6 | MUX6 | R/W | 0h | Selects the output of Mux6 to drive OUTPUT1 of OUTPUT-XBAR<br>0: Respective output of Mux6 is disabled to drive the OUTPUT1 of OUTPUT-XBAR<br>1: Respective output of Mux6 is enabled to drive the OUTPUT1 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5 | MUX5 | R/W | 0h | Selects the output of Mux5 to drive OUTPUT1 of OUTPUT-XBAR<br>0: Respective output of Mux5 is disabled to drive the OUTPUT1 of OUTPUT-XBAR<br>1: Respective output of Mux5 is enabled to drive the OUTPUT1 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 4 | MUX4 | R/W | 0h | Selects the output of Mux4 to drive OUTPUT1 of OUTPUT-XBAR<br>0: Respective output of Mux4 is disabled to drive the OUTPUT1 of OUTPUT-XBAR<br>1: Respective output of Mux4 is enabled to drive the OUTPUT1 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-105. 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<br>0: Respective output of Mux3 is disabled to drive the OUTPUT1 of OUTPUT-XBAR<br>1: Respective output of Mux3 is enabled to drive the OUTPUT1 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 2 | MUX2 | R/W | 0h | Selects the output of Mux2 to drive OUTPUT1 of OUTPUT-XBAR<br>0: Respective output of Mux2 is disabled to drive the OUTPUT1 of OUTPUT-XBAR<br>1: Respective output of Mux2 is enabled to drive the OUTPUT1 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1 | MUX1 | R/W | 0h | Selects the output of Mux1 to drive OUTPUT1 of OUTPUT-XBAR<br>0: Respective output of Mux1 is disabled to drive the OUTPUT1 of OUTPUT-XBAR<br>1: Respective output of Mux1 is enabled to drive the OUTPUT1 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 0 | MUX0 | R/W | 0h | Selects the output of mux0 to drive OUTPUT1 of OUTPUT-XBAR<br>0: Respective output of Mux0 is disabled to drive the OUTPUT1 of OUTPUT-XBAR<br>1: Respective output of Mux0 is enabled to drive the OUTPUT1 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.6.18 OUTPUT2MUXENABLE Register (Offset = 22h) [Reset = 00000000h]

OUTPUT2MUXENABLE is shown in Figure 9-97 and described in Table 9-106.

Return to the Summary Table.

Output X-BAR Mux Enable for Output 2

**Figure 9-97. OUTPUT2MUXENABLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MUX31 | MUX30 | MUX29 | MUX28 | MUX27 | MUX26 | MUX25 | MUX24 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MUX23 | MUX22 | MUX21 | MUX20 | MUX19 | MUX18 | MUX17 | MUX16 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MUX15 | MUX14 | MUX13 | MUX12 | MUX11 | MUX10 | MUX9 | MUX8 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | MUX6 | MUX5 | MUX4 | MUX3 | MUX2 | MUX1 | MUX0 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 9-106. 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<br>0: Respective output of Mux31 is disabled to drive the OUTPUT2 of OUTPUT-XBAR<br>1: Respective output of Mux31 is enabled to drive the OUTPUT2 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 30 | MUX30 | R/W | 0h | Selects the output of Mux30 to drive OUTPUT2 of OUTPUT-XBAR<br>0: Respective output of Mux30 is disabled to drive the OUTPUT2 of OUTPUT-XBAR<br>1: Respective output of Mux30 is enabled to drive the OUTPUT2 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29 | MUX29 | R/W | 0h | Selects the output of Mux29 to drive OUTPUT2 of OUTPUT-XBAR<br>0: Respective output of Mux29 is disabled to drive the OUTPUT2 of OUTPUT-XBAR<br>1: Respective output of Mux29 is enabled to drive the OUTPUT2 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 28 | MUX28 | R/W | 0h | Selects the output of Mux28 to drive OUTPUT2 of OUTPUT-XBAR<br>0: Respective output of Mux28 is disabled to drive the OUTPUT2 of OUTPUT-XBAR<br>1: Respective output of Mux28 is enabled to drive the OUTPUT2 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-106. 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<br>0: Respective output of Mux27 is disabled to drive the OUTPUT2 of OUTPUT-XBAR<br>1: Respective output of Mux27 is enabled to drive the OUTPUT2 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 26 | MUX26 | R/W | 0h | Selects the output of Mux26 to drive OUTPUT2 of OUTPUT-XBAR<br>0: Respective output of Mux26 is disabled to drive the OUTPUT2 of OUTPUT-XBAR<br>1: Respective output of Mux26 is enabled to drive the OUTPUT2 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25 | MUX25 | R/W | 0h | Selects the output of Mux25 to drive OUTPUT2 of OUTPUT-XBAR<br>0: Respective output of Mux25 is disabled to drive the OUTPUT2 of OUTPUT-XBAR<br>1: Respective output of Mux25 is enabled to drive the OUTPUT2 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 24 | MUX24 | R/W | 0h | Selects the output of Mux24 to drive OUTPUT2 of OUTPUT-XBAR<br>0: Respective output of Mux24 is disabled to drive the OUTPUT2 of OUTPUT-XBAR<br>1: Respective output of Mux24 is enabled to drive the OUTPUT2 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23 | MUX23 | R/W | 0h | Selects the output of Mux23 to drive OUTPUT2 of OUTPUT-XBAR<br>0: Respective output of Mux23 is disabled to drive the OUTPUT2 of OUTPUT-XBAR<br>1: Respective output of Mux23 is enabled to drive the OUTPUT2 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 22 | MUX22 | R/W | 0h | Selects the output of Mux22 to drive OUTPUT2 of OUTPUT-XBAR<br>0: Respective output of Mux22 is disabled to drive the OUTPUT2 of OUTPUT-XBAR<br>1: Respective output of Mux22 is enabled to drive the OUTPUT2 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21 | MUX21 | R/W | 0h | Selects the output of Mux21 to drive OUTPUT2 of OUTPUT-XBAR<br>0: Respective output of Mux21 is disabled to drive the OUTPUT2 of OUTPUT-XBAR<br>1: Respective output of Mux21 is enabled to drive the OUTPUT2 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 20 | MUX20 | R/W | 0h | Selects the output of Mux20 to drive OUTPUT2 of OUTPUT-XBAR<br>0: Respective output of Mux20 is disabled to drive the OUTPUT2 of OUTPUT-XBAR<br>1: Respective output of Mux20 is enabled to drive the OUTPUT2 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-106. 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<br>0: Respective output of Mux19 is disabled to drive the OUTPUT2 of OUTPUT-XBAR<br>1: Respective output of Mux19 is enabled to drive the OUTPUT2 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 18 | MUX18 | R/W | 0h | Selects the output of Mux18 to drive OUTPUT2 of OUTPUT-XBAR<br>0: Respective output of Mux18 is disabled to drive the OUTPUT2 of OUTPUT-XBAR<br>1: Respective output of Mux18 is enabled to drive the OUTPUT2 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17 | MUX17 | R/W | 0h | Selects the output of Mux17 to drive OUTPUT2 of OUTPUT-XBAR<br>0: Respective output of Mux17 is disabled to drive the OUTPUT2 of OUTPUT-XBAR<br>1: Respective output of Mux17 is enabled to drive the OUTPUT2 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 16 | MUX16 | R/W | 0h | Selects the output of Mux16 to drive OUTPUT2 of OUTPUT-XBAR<br>0: Respective output of Mux16 is disabled to drive the OUTPUT2 of OUTPUT-XBAR<br>1: Respective output of Mux16 is enabled to drive the OUTPUT2 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15 | MUX15 | R/W | 0h | Selects the output of Mux15 to drive OUTPUT2 of OUTPUT-XBAR<br>0: Respective output of Mux15 is disabled to drive the OUTPUT2 of OUTPUT-XBAR<br>1: Respective output of Mux15 is enabled to drive the OUTPUT2 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 14 | MUX14 | R/W | 0h | Selects the output of Mux14 to drive OUTPUT2 of OUTPUT-XBAR<br>0: Respective output of Mux14 is disabled to drive the OUTPUT2 of OUTPUT-XBAR<br>1: Respective output of Mux14 is enabled to drive the OUTPUT2 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13 | MUX13 | R/W | 0h | Selects the output of Mux13 to drive OUTPUT2 of OUTPUT-XBAR<br>0: Respective output of Mux13 is disabled to drive the OUTPUT2 of OUTPUT-XBAR<br>1: Respective output of Mux13 is enabled to drive the OUTPUT2 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 12 | MUX12 | R/W | 0h | Selects the output of Mux12 to drive OUTPUT2 of OUTPUT-XBAR<br>0: Respective output of Mux12 is disabled to drive the OUTPUT2 of OUTPUT-XBAR<br>1: Respective output of Mux12 is enabled to drive the OUTPUT2 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-106. 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<br>0: Respective output of Mux11 is disabled to drive the OUTPUT2 of OUTPUT-XBAR<br>1: Respective output of Mux11 is enabled to drive the OUTPUT2 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 10 | MUX10 | R/W | 0h | Selects the output of Mux10 to drive OUTPUT2 of OUTPUT-XBAR<br>0: Respective output of Mux10 is disabled to drive the OUTPUT2 of OUTPUT-XBAR<br>1: Respective output of Mux10 is enabled to drive the OUTPUT2 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9 | MUX9 | R/W | 0h | Selects the output of Mux9 to drive OUTPUT2 of OUTPUT-XBAR<br>0: Respective output of Mux9 is disabled to drive the OUTPUT2 of OUTPUT-XBAR<br>1: Respective output of Mux9 is enabled to drive the OUTPUT2 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 8 | MUX8 | R/W | 0h | Selects the output of Mux8 to drive OUTPUT2 of OUTPUT-XBAR<br>0: Respective output of Mux8 is disabled to drive the OUTPUT2 of OUTPUT-XBAR<br>1: Respective output of Mux8 is enabled to drive the OUTPUT2 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7 | MUX7 | R/W | 0h | Selects the output of Mux7 to drive OUTPUT2 of OUTPUT-XBAR<br>0: Respective output of Mux7 is disabled to drive the OUTPUT2 of OUTPUT-XBAR<br>1: Respective output of Mux7 is enabled to drive the OUTPUT2 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 6 | MUX6 | R/W | 0h | Selects the output of Mux6 to drive OUTPUT2 of OUTPUT-XBAR<br>0: Respective output of Mux6 is disabled to drive the OUTPUT2 of OUTPUT-XBAR<br>1: Respective output of Mux6 is enabled to drive the OUTPUT2 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5 | MUX5 | R/W | 0h | Selects the output of Mux5 to drive OUTPUT2 of OUTPUT-XBAR<br>0: Respective output of Mux5 is disabled to drive the OUTPUT2 of OUTPUT-XBAR<br>1: Respective output of Mux5 is enabled to drive the OUTPUT2 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 4 | MUX4 | R/W | 0h | Selects the output of Mux4 to drive OUTPUT2 of OUTPUT-XBAR<br>0: Respective output of Mux4 is disabled to drive the OUTPUT2 of OUTPUT-XBAR<br>1: Respective output of Mux4 is enabled to drive the OUTPUT2 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-106. 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<br>0: Respective output of Mux3 is disabled to drive the OUTPUT2 of OUTPUT-XBAR<br>1: Respective output of Mux3 is enabled to drive the OUTPUT2 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 2 | MUX2 | R/W | 0h | Selects the output of Mux2 to drive OUTPUT2 of OUTPUT-XBAR<br>0: Respective output of Mux2 is disabled to drive the OUTPUT2 of OUTPUT-XBAR<br>1: Respective output of Mux2 is enabled to drive the OUTPUT2 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1 | MUX1 | R/W | 0h | Selects the output of Mux1 to drive OUTPUT2 of OUTPUT-XBAR<br>0: Respective output of Mux1 is disabled to drive the OUTPUT2 of OUTPUT-XBAR<br>1: Respective output of Mux1 is enabled to drive the OUTPUT2 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 0 | MUX0 | R/W | 0h | Selects the output of mux0 to drive OUTPUT2 of OUTPUT-XBAR<br>0: Respective output of Mux0 is disabled to drive the OUTPUT2 of OUTPUT-XBAR<br>1: Respective output of Mux0 is enabled to drive the OUTPUT2 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.6.19 OUTPUT3MUXENABLE Register (Offset = 24h) [Reset = 00000000h]

OUTPUT3MUXENABLE is shown in Figure 9-98 and described in Table 9-107.

Return to the Summary Table.

Output X-BAR Mux Enable for Output 3

**Figure 9-98. OUTPUT3MUXENABLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MUX31 | MUX30 | MUX29 | MUX28 | MUX27 | MUX26 | MUX25 | MUX24 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MUX23 | MUX22 | MUX21 | MUX20 | MUX19 | MUX18 | MUX17 | MUX16 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MUX15 | MUX14 | MUX13 | MUX12 | MUX11 | MUX10 | MUX9 | MUX8 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | MUX6 | MUX5 | MUX4 | MUX3 | MUX2 | MUX1 | MUX0 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 9-107. 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<br>0: Respective output of Mux31 is disabled to drive the OUTPUT3 of OUTPUT-XBAR<br>1: Respective output of Mux31 is enabled to drive the OUTPUT3 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 30 | MUX30 | R/W | 0h | Selects the output of Mux30 to drive OUTPUT3 of OUTPUT-XBAR<br>0: Respective output of Mux30 is disabled to drive the OUTPUT3 of OUTPUT-XBAR<br>1: Respective output of Mux30 is enabled to drive the OUTPUT3 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29 | MUX29 | R/W | 0h | Selects the output of Mux29 to drive OUTPUT3 of OUTPUT-XBAR<br>0: Respective output of Mux29 is disabled to drive the OUTPUT3 of OUTPUT-XBAR<br>1: Respective output of Mux29 is enabled to drive the OUTPUT3 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 28 | MUX28 | R/W | 0h | Selects the output of Mux28 to drive OUTPUT3 of OUTPUT-XBAR<br>0: Respective output of Mux28 is disabled to drive the OUTPUT3 of OUTPUT-XBAR<br>1: Respective output of Mux28 is enabled to drive the OUTPUT3 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-107. 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<br>0: Respective output of Mux27 is disabled to drive the OUTPUT3 of OUTPUT-XBAR<br>1: Respective output of Mux27 is enabled to drive the OUTPUT3 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 26 | MUX26 | R/W | 0h | Selects the output of Mux26 to drive OUTPUT3 of OUTPUT-XBAR<br>0: Respective output of Mux26 is disabled to drive the OUTPUT3 of OUTPUT-XBAR<br>1: Respective output of Mux26 is enabled to drive the OUTPUT3 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25 | MUX25 | R/W | 0h | Selects the output of Mux25 to drive OUTPUT3 of OUTPUT-XBAR<br>0: Respective output of Mux25 is disabled to drive the OUTPUT3 of OUTPUT-XBAR<br>1: Respective output of Mux25 is enabled to drive the OUTPUT3 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 24 | MUX24 | R/W | 0h | Selects the output of Mux24 to drive OUTPUT3 of OUTPUT-XBAR<br>0: Respective output of Mux24 is disabled to drive the OUTPUT3 of OUTPUT-XBAR<br>1: Respective output of Mux24 is enabled to drive the OUTPUT3 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23 | MUX23 | R/W | 0h | Selects the output of Mux23 to drive OUTPUT3 of OUTPUT-XBAR<br>0: Respective output of Mux23 is disabled to drive the OUTPUT3 of OUTPUT-XBAR<br>1: Respective output of Mux23 is enabled to drive the OUTPUT3 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 22 | MUX22 | R/W | 0h | Selects the output of Mux22 to drive OUTPUT3 of OUTPUT-XBAR<br>0: Respective output of Mux22 is disabled to drive the OUTPUT3 of OUTPUT-XBAR<br>1: Respective output of Mux22 is enabled to drive the OUTPUT3 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21 | MUX21 | R/W | 0h | Selects the output of Mux21 to drive OUTPUT3 of OUTPUT-XBAR<br>0: Respective output of Mux21 is disabled to drive the OUTPUT3 of OUTPUT-XBAR<br>1: Respective output of Mux21 is enabled to drive the OUTPUT3 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 20 | MUX20 | R/W | 0h | Selects the output of Mux20 to drive OUTPUT3 of OUTPUT-XBAR<br>0: Respective output of Mux20 is disabled to drive the OUTPUT3 of OUTPUT-XBAR<br>1: Respective output of Mux20 is enabled to drive the OUTPUT3 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-107. 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<br>0: Respective output of Mux19 is disabled to drive the OUTPUT3 of OUTPUT-XBAR<br>1: Respective output of Mux19 is enabled to drive the OUTPUT3 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 18 | MUX18 | R/W | 0h | Selects the output of Mux18 to drive OUTPUT3 of OUTPUT-XBAR<br>0: Respective output of Mux18 is disabled to drive the OUTPUT3 of OUTPUT-XBAR<br>1: Respective output of Mux18 is enabled to drive the OUTPUT3 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17 | MUX17 | R/W | 0h | Selects the output of Mux17 to drive OUTPUT3 of OUTPUT-XBAR<br>0: Respective output of Mux17 is disabled to drive the OUTPUT3 of OUTPUT-XBAR<br>1: Respective output of Mux17 is enabled to drive the OUTPUT3 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 16 | MUX16 | R/W | 0h | Selects the output of Mux16 to drive OUTPUT3 of OUTPUT-XBAR<br>0: Respective output of Mux16 is disabled to drive the OUTPUT3 of OUTPUT-XBAR<br>1: Respective output of Mux16 is enabled to drive the OUTPUT3 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15 | MUX15 | R/W | 0h | Selects the output of Mux15 to drive OUTPUT3 of OUTPUT-XBAR<br>0: Respective output of Mux15 is disabled to drive the OUTPUT3 of OUTPUT-XBAR<br>1: Respective output of Mux15 is enabled to drive the OUTPUT3 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 14 | MUX14 | R/W | 0h | Selects the output of Mux14 to drive OUTPUT3 of OUTPUT-XBAR<br>0: Respective output of Mux14 is disabled to drive the OUTPUT3 of OUTPUT-XBAR<br>1: Respective output of Mux14 is enabled to drive the OUTPUT3 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13 | MUX13 | R/W | 0h | Selects the output of Mux13 to drive OUTPUT3 of OUTPUT-XBAR<br>0: Respective output of Mux13 is disabled to drive the OUTPUT3 of OUTPUT-XBAR<br>1: Respective output of Mux13 is enabled to drive the OUTPUT3 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 12 | MUX12 | R/W | 0h | Selects the output of Mux12 to drive OUTPUT3 of OUTPUT-XBAR<br>0: Respective output of Mux12 is disabled to drive the OUTPUT3 of OUTPUT-XBAR<br>1: Respective output of Mux12 is enabled to drive the OUTPUT3 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-107. 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<br>0: Respective output of Mux11 is disabled to drive the OUTPUT3 of OUTPUT-XBAR<br>1: Respective output of Mux11 is enabled to drive the OUTPUT3 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 10 | MUX10 | R/W | 0h | Selects the output of Mux10 to drive OUTPUT3 of OUTPUT-XBAR<br>0: Respective output of Mux10 is disabled to drive the OUTPUT3 of OUTPUT-XBAR<br>1: Respective output of Mux10 is enabled to drive the OUTPUT3 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9 | MUX9 | R/W | 0h | Selects the output of Mux9 to drive OUTPUT3 of OUTPUT-XBAR<br>0: Respective output of Mux9 is disabled to drive the OUTPUT3 of OUTPUT-XBAR<br>1: Respective output of Mux9 is enabled to drive the OUTPUT3 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 8 | MUX8 | R/W | 0h | Selects the output of Mux8 to drive OUTPUT3 of OUTPUT-XBAR<br>0: Respective output of Mux8 is disabled to drive the OUTPUT3 of OUTPUT-XBAR<br>1: Respective output of Mux8 is enabled to drive the OUTPUT3 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7 | MUX7 | R/W | 0h | Selects the output of Mux7 to drive OUTPUT3 of OUTPUT-XBAR<br>0: Respective output of Mux7 is disabled to drive the OUTPUT3 of OUTPUT-XBAR<br>1: Respective output of Mux7 is enabled to drive the OUTPUT3 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 6 | MUX6 | R/W | 0h | Selects the output of Mux6 to drive OUTPUT3 of OUTPUT-XBAR<br>0: Respective output of Mux6 is disabled to drive the OUTPUT3 of OUTPUT-XBAR<br>1: Respective output of Mux6 is enabled to drive the OUTPUT3 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5 | MUX5 | R/W | 0h | Selects the output of Mux5 to drive OUTPUT3 of OUTPUT-XBAR<br>0: Respective output of Mux5 is disabled to drive the OUTPUT3 of OUTPUT-XBAR<br>1: Respective output of Mux5 is enabled to drive the OUTPUT3 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 4 | MUX4 | R/W | 0h | Selects the output of Mux4 to drive OUTPUT3 of OUTPUT-XBAR<br>0: Respective output of Mux4 is disabled to drive the OUTPUT3 of OUTPUT-XBAR<br>1: Respective output of Mux4 is enabled to drive the OUTPUT3 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-107. 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<br>0: Respective output of Mux3 is disabled to drive the OUTPUT3 of OUTPUT-XBAR<br>1: Respective output of Mux3 is enabled to drive the OUTPUT3 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 2 | MUX2 | R/W | 0h | Selects the output of Mux2 to drive OUTPUT3 of OUTPUT-XBAR<br>0: Respective output of Mux2 is disabled to drive the OUTPUT3 of OUTPUT-XBAR<br>1: Respective output of Mux2 is enabled to drive the OUTPUT3 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1 | MUX1 | R/W | 0h | Selects the output of Mux1 to drive OUTPUT3 of OUTPUT-XBAR<br>0: Respective output of Mux1 is disabled to drive the OUTPUT3 of OUTPUT-XBAR<br>1: Respective output of Mux1 is enabled to drive the OUTPUT3 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 0 | MUX0 | R/W | 0h | Selects the output of mux0 to drive OUTPUT3 of OUTPUT-XBAR<br>0: Respective output of Mux0 is disabled to drive the OUTPUT3 of OUTPUT-XBAR<br>1: Respective output of Mux0 is enabled to drive the OUTPUT3 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.6.20 OUTPUT4MUXENABLE Register (Offset = 26h) [Reset = 00000000h]

OUTPUT4MUXENABLE is shown in Figure 9-99 and described in Table 9-108.

Return to the Summary Table.

Output X-BAR Mux Enable for Output 4

**Figure 9-99. OUTPUT4MUXENABLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MUX31 | MUX30 | MUX29 | MUX28 | MUX27 | MUX26 | MUX25 | MUX24 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MUX23 | MUX22 | MUX21 | MUX20 | MUX19 | MUX18 | MUX17 | MUX16 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MUX15 | MUX14 | MUX13 | MUX12 | MUX11 | MUX10 | MUX9 | MUX8 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | MUX6 | MUX5 | MUX4 | MUX3 | MUX2 | MUX1 | MUX0 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 9-108. 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<br>0: Respective output of Mux31 is disabled to drive the OUTPUT4 of OUTPUT-XBAR<br>1: Respective output of Mux31 is enabled to drive the OUTPUT4 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 30 | MUX30 | R/W | 0h | Selects the output of Mux30 to drive OUTPUT4 of OUTPUT-XBAR<br>0: Respective output of Mux30 is disabled to drive the OUTPUT4 of OUTPUT-XBAR<br>1: Respective output of Mux30 is enabled to drive the OUTPUT4 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29 | MUX29 | R/W | 0h | Selects the output of Mux29 to drive OUTPUT4 of OUTPUT-XBAR<br>0: Respective output of Mux29 is disabled to drive the OUTPUT4 of OUTPUT-XBAR<br>1: Respective output of Mux29 is enabled to drive the OUTPUT4 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 28 | MUX28 | R/W | 0h | Selects the output of Mux28 to drive OUTPUT4 of OUTPUT-XBAR<br>0: Respective output of Mux28 is disabled to drive the OUTPUT4 of OUTPUT-XBAR<br>1: Respective output of Mux28 is enabled to drive the OUTPUT4 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-108. 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<br>0: Respective output of Mux27 is disabled to drive the OUTPUT4 of OUTPUT-XBAR<br>1: Respective output of Mux27 is enabled to drive the OUTPUT4 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 26 | MUX26 | R/W | 0h | Selects the output of Mux26 to drive OUTPUT4 of OUTPUT-XBAR<br>0: Respective output of Mux26 is disabled to drive the OUTPUT4 of OUTPUT-XBAR<br>1: Respective output of Mux26 is enabled to drive the OUTPUT4 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25 | MUX25 | R/W | 0h | Selects the output of Mux25 to drive OUTPUT4 of OUTPUT-XBAR<br>0: Respective output of Mux25 is disabled to drive the OUTPUT4 of OUTPUT-XBAR<br>1: Respective output of Mux25 is enabled to drive the OUTPUT4 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 24 | MUX24 | R/W | 0h | Selects the output of Mux24 to drive OUTPUT4 of OUTPUT-XBAR<br>0: Respective output of Mux24 is disabled to drive the OUTPUT4 of OUTPUT-XBAR<br>1: Respective output of Mux24 is enabled to drive the OUTPUT4 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23 | MUX23 | R/W | 0h | Selects the output of Mux23 to drive OUTPUT4 of OUTPUT-XBAR<br>0: Respective output of Mux23 is disabled to drive the OUTPUT4 of OUTPUT-XBAR<br>1: Respective output of Mux23 is enabled to drive the OUTPUT4 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 22 | MUX22 | R/W | 0h | Selects the output of Mux22 to drive OUTPUT4 of OUTPUT-XBAR<br>0: Respective output of Mux22 is disabled to drive the OUTPUT4 of OUTPUT-XBAR<br>1: Respective output of Mux22 is enabled to drive the OUTPUT4 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21 | MUX21 | R/W | 0h | Selects the output of Mux21 to drive OUTPUT4 of OUTPUT-XBAR<br>0: Respective output of Mux21 is disabled to drive the OUTPUT4 of OUTPUT-XBAR<br>1: Respective output of Mux21 is enabled to drive the OUTPUT4 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 20 | MUX20 | R/W | 0h | Selects the output of Mux20 to drive OUTPUT4 of OUTPUT-XBAR<br>0: Respective output of Mux20 is disabled to drive the OUTPUT4 of OUTPUT-XBAR<br>1: Respective output of Mux20 is enabled to drive the OUTPUT4 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-108. 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<br>0: Respective output of Mux19 is disabled to drive the OUTPUT4 of OUTPUT-XBAR<br>1: Respective output of Mux19 is enabled to drive the OUTPUT4 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 18 | MUX18 | R/W | 0h | Selects the output of Mux18 to drive OUTPUT4 of OUTPUT-XBAR<br>0: Respective output of Mux18 is disabled to drive the OUTPUT4 of OUTPUT-XBAR<br>1: Respective output of Mux18 is enabled to drive the OUTPUT4 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17 | MUX17 | R/W | 0h | Selects the output of Mux17 to drive OUTPUT4 of OUTPUT-XBAR<br>0: Respective output of Mux17 is disabled to drive the OUTPUT4 of OUTPUT-XBAR<br>1: Respective output of Mux17 is enabled to drive the OUTPUT4 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 16 | MUX16 | R/W | 0h | Selects the output of Mux16 to drive OUTPUT4 of OUTPUT-XBAR<br>0: Respective output of Mux16 is disabled to drive the OUTPUT4 of OUTPUT-XBAR<br>1: Respective output of Mux16 is enabled to drive the OUTPUT4 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15 | MUX15 | R/W | 0h | Selects the output of Mux15 to drive OUTPUT4 of OUTPUT-XBAR<br>0: Respective output of Mux15 is disabled to drive the OUTPUT4 of OUTPUT-XBAR<br>1: Respective output of Mux15 is enabled to drive the OUTPUT4 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 14 | MUX14 | R/W | 0h | Selects the output of Mux14 to drive OUTPUT4 of OUTPUT-XBAR<br>0: Respective output of Mux14 is disabled to drive the OUTPUT4 of OUTPUT-XBAR<br>1: Respective output of Mux14 is enabled to drive the OUTPUT4 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13 | MUX13 | R/W | 0h | Selects the output of Mux13 to drive OUTPUT4 of OUTPUT-XBAR<br>0: Respective output of Mux13 is disabled to drive the OUTPUT4 of OUTPUT-XBAR<br>1: Respective output of Mux13 is enabled to drive the OUTPUT4 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 12 | MUX12 | R/W | 0h | Selects the output of Mux12 to drive OUTPUT4 of OUTPUT-XBAR<br>0: Respective output of Mux12 is disabled to drive the OUTPUT4 of OUTPUT-XBAR<br>1: Respective output of Mux12 is enabled to drive the OUTPUT4 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-108. 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<br>0: Respective output of Mux11 is disabled to drive the OUTPUT4 of OUTPUT-XBAR<br>1: Respective output of Mux11 is enabled to drive the OUTPUT4 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 10 | MUX10 | R/W | 0h | Selects the output of Mux10 to drive OUTPUT4 of OUTPUT-XBAR<br>0: Respective output of Mux10 is disabled to drive the OUTPUT4 of OUTPUT-XBAR<br>1: Respective output of Mux10 is enabled to drive the OUTPUT4 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9 | MUX9 | R/W | 0h | Selects the output of Mux9 to drive OUTPUT4 of OUTPUT-XBAR<br>0: Respective output of Mux9 is disabled to drive the OUTPUT4 of OUTPUT-XBAR<br>1: Respective output of Mux9 is enabled to drive the OUTPUT4 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 8 | MUX8 | R/W | 0h | Selects the output of Mux8 to drive OUTPUT4 of OUTPUT-XBAR<br>0: Respective output of Mux8 is disabled to drive the OUTPUT4 of OUTPUT-XBAR<br>1: Respective output of Mux8 is enabled to drive the OUTPUT4 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7 | MUX7 | R/W | 0h | Selects the output of Mux7 to drive OUTPUT4 of OUTPUT-XBAR<br>0: Respective output of Mux7 is disabled to drive the OUTPUT4 of OUTPUT-XBAR<br>1: Respective output of Mux7 is enabled to drive the OUTPUT4 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 6 | MUX6 | R/W | 0h | Selects the output of Mux6 to drive OUTPUT4 of OUTPUT-XBAR<br>0: Respective output of Mux6 is disabled to drive the OUTPUT4 of OUTPUT-XBAR<br>1: Respective output of Mux6 is enabled to drive the OUTPUT4 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5 | MUX5 | R/W | 0h | Selects the output of Mux5 to drive OUTPUT4 of OUTPUT-XBAR<br>0: Respective output of Mux5 is disabled to drive the OUTPUT4 of OUTPUT-XBAR<br>1: Respective output of Mux5 is enabled to drive the OUTPUT4 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 4 | MUX4 | R/W | 0h | Selects the output of Mux4 to drive OUTPUT4 of OUTPUT-XBAR<br>0: Respective output of Mux4 is disabled to drive the OUTPUT4 of OUTPUT-XBAR<br>1: Respective output of Mux4 is enabled to drive the OUTPUT4 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-108. 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<br>0: Respective output of Mux3 is disabled to drive the OUTPUT4 of OUTPUT-XBAR<br>1: Respective output of Mux3 is enabled to drive the OUTPUT4 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 2 | MUX2 | R/W | 0h | Selects the output of Mux2 to drive OUTPUT4 of OUTPUT-XBAR<br>0: Respective output of Mux2 is disabled to drive the OUTPUT4 of OUTPUT-XBAR<br>1: Respective output of Mux2 is enabled to drive the OUTPUT4 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1 | MUX1 | R/W | 0h | Selects the output of Mux1 to drive OUTPUT4 of OUTPUT-XBAR<br>0: Respective output of Mux1 is disabled to drive the OUTPUT4 of OUTPUT-XBAR<br>1: Respective output of Mux1 is enabled to drive the OUTPUT4 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 0 | MUX0 | R/W | 0h | Selects the output of mux0 to drive OUTPUT4 of OUTPUT-XBAR<br>0: Respective output of Mux0 is disabled to drive the OUTPUT4 of OUTPUT-XBAR<br>1: Respective output of Mux0 is enabled to drive the OUTPUT4 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.6.21 OUTPUT5MUXENABLE Register (Offset = 28h) [Reset = 00000000h]

OUTPUT5MUXENABLE is shown in Figure 9-100 and described in Table 9-109.

Return to the Summary Table.

Output X-BAR Mux Enable for Output 5

**Figure 9-100. OUTPUT5MUXENABLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MUX31 | MUX30 | MUX29 | MUX28 | MUX27 | MUX26 | MUX25 | MUX24 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MUX23 | MUX22 | MUX21 | MUX20 | MUX19 | MUX18 | MUX17 | MUX16 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MUX15 | MUX14 | MUX13 | MUX12 | MUX11 | MUX10 | MUX9 | MUX8 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | MUX6 | MUX5 | MUX4 | MUX3 | MUX2 | MUX1 | MUX0 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 9-109. 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<br>0: Respective output of Mux31 is disabled to drive the OUTPUT5 of OUTPUT-XBAR<br>1: Respective output of Mux31 is enabled to drive the OUTPUT5 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 30 | MUX30 | R/W | 0h | Selects the output of Mux30 to drive OUTPUT5 of OUTPUT-XBAR<br>0: Respective output of Mux30 is disabled to drive the OUTPUT5 of OUTPUT-XBAR<br>1: Respective output of Mux30 is enabled to drive the OUTPUT5 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29 | MUX29 | R/W | 0h | Selects the output of Mux29 to drive OUTPUT5 of OUTPUT-XBAR<br>0: Respective output of Mux29 is disabled to drive the OUTPUT5 of OUTPUT-XBAR<br>1: Respective output of Mux29 is enabled to drive the OUTPUT5 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 28 | MUX28 | R/W | 0h | Selects the output of Mux28 to drive OUTPUT5 of OUTPUT-XBAR<br>0: Respective output of Mux28 is disabled to drive the OUTPUT5 of OUTPUT-XBAR<br>1: Respective output of Mux28 is enabled to drive the OUTPUT5 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-109. 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<br>0: Respective output of Mux27 is disabled to drive the OUTPUT5 of OUTPUT-XBAR<br>1: Respective output of Mux27 is enabled to drive the OUTPUT5 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 26 | MUX26 | R/W | 0h | Selects the output of Mux26 to drive OUTPUT5 of OUTPUT-XBAR<br>0: Respective output of Mux26 is disabled to drive the OUTPUT5 of OUTPUT-XBAR<br>1: Respective output of Mux26 is enabled to drive the OUTPUT5 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25 | MUX25 | R/W | 0h | Selects the output of Mux25 to drive OUTPUT5 of OUTPUT-XBAR<br>0: Respective output of Mux25 is disabled to drive the OUTPUT5 of OUTPUT-XBAR<br>1: Respective output of Mux25 is enabled to drive the OUTPUT5 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 24 | MUX24 | R/W | 0h | Selects the output of Mux24 to drive OUTPUT5 of OUTPUT-XBAR<br>0: Respective output of Mux24 is disabled to drive the OUTPUT5 of OUTPUT-XBAR<br>1: Respective output of Mux24 is enabled to drive the OUTPUT5 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23 | MUX23 | R/W | 0h | Selects the output of Mux23 to drive OUTPUT5 of OUTPUT-XBAR<br>0: Respective output of Mux23 is disabled to drive the OUTPUT5 of OUTPUT-XBAR<br>1: Respective output of Mux23 is enabled to drive the OUTPUT5 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 22 | MUX22 | R/W | 0h | Selects the output of Mux22 to drive OUTPUT5 of OUTPUT-XBAR<br>0: Respective output of Mux22 is disabled to drive the OUTPUT5 of OUTPUT-XBAR<br>1: Respective output of Mux22 is enabled to drive the OUTPUT5 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21 | MUX21 | R/W | 0h | Selects the output of Mux21 to drive OUTPUT5 of OUTPUT-XBAR<br>0: Respective output of Mux21 is disabled to drive the OUTPUT5 of OUTPUT-XBAR<br>1: Respective output of Mux21 is enabled to drive the OUTPUT5 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 20 | MUX20 | R/W | 0h | Selects the output of Mux20 to drive OUTPUT5 of OUTPUT-XBAR<br>0: Respective output of Mux20 is disabled to drive the OUTPUT5 of OUTPUT-XBAR<br>1: Respective output of Mux20 is enabled to drive the OUTPUT5 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-109. 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<br>0: Respective output of Mux19 is disabled to drive the OUTPUT5 of OUTPUT-XBAR<br>1: Respective output of Mux19 is enabled to drive the OUTPUT5 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 18 | MUX18 | R/W | 0h | Selects the output of Mux18 to drive OUTPUT5 of OUTPUT-XBAR<br>0: Respective output of Mux18 is disabled to drive the OUTPUT5 of OUTPUT-XBAR<br>1: Respective output of Mux18 is enabled to drive the OUTPUT5 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17 | MUX17 | R/W | 0h | Selects the output of Mux17 to drive OUTPUT5 of OUTPUT-XBAR<br>0: Respective output of Mux17 is disabled to drive the OUTPUT5 of OUTPUT-XBAR<br>1: Respective output of Mux17 is enabled to drive the OUTPUT5 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 16 | MUX16 | R/W | 0h | Selects the output of Mux16 to drive OUTPUT5 of OUTPUT-XBAR<br>0: Respective output of Mux16 is disabled to drive the OUTPUT5 of OUTPUT-XBAR<br>1: Respective output of Mux16 is enabled to drive the OUTPUT5 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15 | MUX15 | R/W | 0h | Selects the output of Mux15 to drive OUTPUT5 of OUTPUT-XBAR<br>0: Respective output of Mux15 is disabled to drive the OUTPUT5 of OUTPUT-XBAR<br>1: Respective output of Mux15 is enabled to drive the OUTPUT5 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 14 | MUX14 | R/W | 0h | Selects the output of Mux14 to drive OUTPUT5 of OUTPUT-XBAR<br>0: Respective output of Mux14 is disabled to drive the OUTPUT5 of OUTPUT-XBAR<br>1: Respective output of Mux14 is enabled to drive the OUTPUT5 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13 | MUX13 | R/W | 0h | Selects the output of Mux13 to drive OUTPUT5 of OUTPUT-XBAR<br>0: Respective output of Mux13 is disabled to drive the OUTPUT5 of OUTPUT-XBAR<br>1: Respective output of Mux13 is enabled to drive the OUTPUT5 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 12 | MUX12 | R/W | 0h | Selects the output of Mux12 to drive OUTPUT5 of OUTPUT-XBAR<br>0: Respective output of Mux12 is disabled to drive the OUTPUT5 of OUTPUT-XBAR<br>1: Respective output of Mux12 is enabled to drive the OUTPUT5 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-109. 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<br>0: Respective output of Mux11 is disabled to drive the OUTPUT5 of OUTPUT-XBAR<br>1: Respective output of Mux11 is enabled to drive the OUTPUT5 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 10 | MUX10 | R/W | 0h | Selects the output of Mux10 to drive OUTPUT5 of OUTPUT-XBAR<br>0: Respective output of Mux10 is disabled to drive the OUTPUT5 of OUTPUT-XBAR<br>1: Respective output of Mux10 is enabled to drive the OUTPUT5 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9 | MUX9 | R/W | 0h | Selects the output of Mux9 to drive OUTPUT5 of OUTPUT-XBAR<br>0: Respective output of Mux9 is disabled to drive the OUTPUT5 of OUTPUT-XBAR<br>1: Respective output of Mux9 is enabled to drive the OUTPUT5 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 8 | MUX8 | R/W | 0h | Selects the output of Mux8 to drive OUTPUT5 of OUTPUT-XBAR<br>0: Respective output of Mux8 is disabled to drive the OUTPUT5 of OUTPUT-XBAR<br>1: Respective output of Mux8 is enabled to drive the OUTPUT5 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7 | MUX7 | R/W | 0h | Selects the output of Mux7 to drive OUTPUT5 of OUTPUT-XBAR<br>0: Respective output of Mux7 is disabled to drive the OUTPUT5 of OUTPUT-XBAR<br>1: Respective output of Mux7 is enabled to drive the OUTPUT5 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 6 | MUX6 | R/W | 0h | Selects the output of Mux6 to drive OUTPUT5 of OUTPUT-XBAR<br>0: Respective output of Mux6 is disabled to drive the OUTPUT5 of OUTPUT-XBAR<br>1: Respective output of Mux6 is enabled to drive the OUTPUT5 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5 | MUX5 | R/W | 0h | Selects the output of Mux5 to drive OUTPUT5 of OUTPUT-XBAR<br>0: Respective output of Mux5 is disabled to drive the OUTPUT5 of OUTPUT-XBAR<br>1: Respective output of Mux5 is enabled to drive the OUTPUT5 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 4 | MUX4 | R/W | 0h | Selects the output of Mux4 to drive OUTPUT5 of OUTPUT-XBAR<br>0: Respective output of Mux4 is disabled to drive the OUTPUT5 of OUTPUT-XBAR<br>1: Respective output of Mux4 is enabled to drive the OUTPUT5 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-109. 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<br>0: Respective output of Mux3 is disabled to drive the OUTPUT5 of OUTPUT-XBAR<br>1: Respective output of Mux3 is enabled to drive the OUTPUT5 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 2 | MUX2 | R/W | 0h | Selects the output of Mux2 to drive OUTPUT5 of OUTPUT-XBAR<br>0: Respective output of Mux2 is disabled to drive the OUTPUT5 of OUTPUT-XBAR<br>1: Respective output of Mux2 is enabled to drive the OUTPUT5 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1 | MUX1 | R/W | 0h | Selects the output of Mux1 to drive OUTPUT5 of OUTPUT-XBAR<br>0: Respective output of Mux1 is disabled to drive the OUTPUT5 of OUTPUT-XBAR<br>1: Respective output of Mux1 is enabled to drive the OUTPUT5 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 0 | MUX0 | R/W | 0h | Selects the output of mux0 to drive OUTPUT5 of OUTPUT-XBAR<br>0: Respective output of Mux0 is disabled to drive the OUTPUT5 of OUTPUT-XBAR<br>1: Respective output of Mux0 is enabled to drive the OUTPUT5 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.6.22 OUTPUT6MUXENABLE Register (Offset = 2Ah) [Reset = 00000000h]

OUTPUT6MUXENABLE is shown in Figure 9-101 and described in Table 9-110.

Return to the Summary Table.

Output X-BAR Mux Enable for Output 6

**Figure 9-101. OUTPUT6MUXENABLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MUX31 | MUX30 | MUX29 | MUX28 | MUX27 | MUX26 | MUX25 | MUX24 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MUX23 | MUX22 | MUX21 | MUX20 | MUX19 | MUX18 | MUX17 | MUX16 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MUX15 | MUX14 | MUX13 | MUX12 | MUX11 | MUX10 | MUX9 | MUX8 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | MUX6 | MUX5 | MUX4 | MUX3 | MUX2 | MUX1 | MUX0 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 9-110. 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<br>0: Respective output of Mux31 is disabled to drive the OUTPUT6 of OUTPUT-XBAR<br>1: Respective output of Mux31 is enabled to drive the OUTPUT6 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 30 | MUX30 | R/W | 0h | Selects the output of Mux30 to drive OUTPUT6 of OUTPUT-XBAR<br>0: Respective output of Mux30 is disabled to drive the OUTPUT6 of OUTPUT-XBAR<br>1: Respective output of Mux30 is enabled to drive the OUTPUT6 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29 | MUX29 | R/W | 0h | Selects the output of Mux29 to drive OUTPUT6 of OUTPUT-XBAR<br>0: Respective output of Mux29 is disabled to drive the OUTPUT6 of OUTPUT-XBAR<br>1: Respective output of Mux29 is enabled to drive the OUTPUT6 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 28 | MUX28 | R/W | 0h | Selects the output of Mux28 to drive OUTPUT6 of OUTPUT-XBAR<br>0: Respective output of Mux28 is disabled to drive the OUTPUT6 of OUTPUT-XBAR<br>1: Respective output of Mux28 is enabled to drive the OUTPUT6 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-110. 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<br>0: Respective output of Mux27 is disabled to drive the OUTPUT6 of OUTPUT-XBAR<br>1: Respective output of Mux27 is enabled to drive the OUTPUT6 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 26 | MUX26 | R/W | 0h | Selects the output of Mux26 to drive OUTPUT6 of OUTPUT-XBAR<br>0: Respective output of Mux26 is disabled to drive the OUTPUT6 of OUTPUT-XBAR<br>1: Respective output of Mux26 is enabled to drive the OUTPUT6 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25 | MUX25 | R/W | 0h | Selects the output of Mux25 to drive OUTPUT6 of OUTPUT-XBAR<br>0: Respective output of Mux25 is disabled to drive the OUTPUT6 of OUTPUT-XBAR<br>1: Respective output of Mux25 is enabled to drive the OUTPUT6 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 24 | MUX24 | R/W | 0h | Selects the output of Mux24 to drive OUTPUT6 of OUTPUT-XBAR<br>0: Respective output of Mux24 is disabled to drive the OUTPUT6 of OUTPUT-XBAR<br>1: Respective output of Mux24 is enabled to drive the OUTPUT6 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23 | MUX23 | R/W | 0h | Selects the output of Mux23 to drive OUTPUT6 of OUTPUT-XBAR<br>0: Respective output of Mux23 is disabled to drive the OUTPUT6 of OUTPUT-XBAR<br>1: Respective output of Mux23 is enabled to drive the OUTPUT6 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 22 | MUX22 | R/W | 0h | Selects the output of Mux22 to drive OUTPUT6 of OUTPUT-XBAR<br>0: Respective output of Mux22 is disabled to drive the OUTPUT6 of OUTPUT-XBAR<br>1: Respective output of Mux22 is enabled to drive the OUTPUT6 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21 | MUX21 | R/W | 0h | Selects the output of Mux21 to drive OUTPUT6 of OUTPUT-XBAR<br>0: Respective output of Mux21 is disabled to drive the OUTPUT6 of OUTPUT-XBAR<br>1: Respective output of Mux21 is enabled to drive the OUTPUT6 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 20 | MUX20 | R/W | 0h | Selects the output of Mux20 to drive OUTPUT6 of OUTPUT-XBAR<br>0: Respective output of Mux20 is disabled to drive the OUTPUT6 of OUTPUT-XBAR<br>1: Respective output of Mux20 is enabled to drive the OUTPUT6 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-110. 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<br>0: Respective output of Mux19 is disabled to drive the OUTPUT6 of OUTPUT-XBAR<br>1: Respective output of Mux19 is enabled to drive the OUTPUT6 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 18 | MUX18 | R/W | 0h | Selects the output of Mux18 to drive OUTPUT6 of OUTPUT-XBAR<br>0: Respective output of Mux18 is disabled to drive the OUTPUT6 of OUTPUT-XBAR<br>1: Respective output of Mux18 is enabled to drive the OUTPUT6 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17 | MUX17 | R/W | 0h | Selects the output of Mux17 to drive OUTPUT6 of OUTPUT-XBAR<br>0: Respective output of Mux17 is disabled to drive the OUTPUT6 of OUTPUT-XBAR<br>1: Respective output of Mux17 is enabled to drive the OUTPUT6 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 16 | MUX16 | R/W | 0h | Selects the output of Mux16 to drive OUTPUT6 of OUTPUT-XBAR<br>0: Respective output of Mux16 is disabled to drive the OUTPUT6 of OUTPUT-XBAR<br>1: Respective output of Mux16 is enabled to drive the OUTPUT6 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15 | MUX15 | R/W | 0h | Selects the output of Mux15 to drive OUTPUT6 of OUTPUT-XBAR<br>0: Respective output of Mux15 is disabled to drive the OUTPUT6 of OUTPUT-XBAR<br>1: Respective output of Mux15 is enabled to drive the OUTPUT6 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 14 | MUX14 | R/W | 0h | Selects the output of Mux14 to drive OUTPUT6 of OUTPUT-XBAR<br>0: Respective output of Mux14 is disabled to drive the OUTPUT6 of OUTPUT-XBAR<br>1: Respective output of Mux14 is enabled to drive the OUTPUT6 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13 | MUX13 | R/W | 0h | Selects the output of Mux13 to drive OUTPUT6 of OUTPUT-XBAR<br>0: Respective output of Mux13 is disabled to drive the OUTPUT6 of OUTPUT-XBAR<br>1: Respective output of Mux13 is enabled to drive the OUTPUT6 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 12 | MUX12 | R/W | 0h | Selects the output of Mux12 to drive OUTPUT6 of OUTPUT-XBAR<br>0: Respective output of Mux12 is disabled to drive the OUTPUT6 of OUTPUT-XBAR<br>1: Respective output of Mux12 is enabled to drive the OUTPUT6 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-110. 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<br>0: Respective output of Mux11 is disabled to drive the OUTPUT6 of OUTPUT-XBAR<br>1: Respective output of Mux11 is enabled to drive the OUTPUT6 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 10 | MUX10 | R/W | 0h | Selects the output of Mux10 to drive OUTPUT6 of OUTPUT-XBAR<br>0: Respective output of Mux10 is disabled to drive the OUTPUT6 of OUTPUT-XBAR<br>1: Respective output of Mux10 is enabled to drive the OUTPUT6 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9 | MUX9 | R/W | 0h | Selects the output of Mux9 to drive OUTPUT6 of OUTPUT-XBAR<br>0: Respective output of Mux9 is disabled to drive the OUTPUT6 of OUTPUT-XBAR<br>1: Respective output of Mux9 is enabled to drive the OUTPUT6 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 8 | MUX8 | R/W | 0h | Selects the output of Mux8 to drive OUTPUT6 of OUTPUT-XBAR<br>0: Respective output of Mux8 is disabled to drive the OUTPUT6 of OUTPUT-XBAR<br>1: Respective output of Mux8 is enabled to drive the OUTPUT6 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7 | MUX7 | R/W | 0h | Selects the output of Mux7 to drive OUTPUT6 of OUTPUT-XBAR<br>0: Respective output of Mux7 is disabled to drive the OUTPUT6 of OUTPUT-XBAR<br>1: Respective output of Mux7 is enabled to drive the OUTPUT6 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 6 | MUX6 | R/W | 0h | Selects the output of Mux6 to drive OUTPUT6 of OUTPUT-XBAR<br>0: Respective output of Mux6 is disabled to drive the OUTPUT6 of OUTPUT-XBAR<br>1: Respective output of Mux6 is enabled to drive the OUTPUT6 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5 | MUX5 | R/W | 0h | Selects the output of Mux5 to drive OUTPUT6 of OUTPUT-XBAR<br>0: Respective output of Mux5 is disabled to drive the OUTPUT6 of OUTPUT-XBAR<br>1: Respective output of Mux5 is enabled to drive the OUTPUT6 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 4 | MUX4 | R/W | 0h | Selects the output of Mux4 to drive OUTPUT6 of OUTPUT-XBAR<br>0: Respective output of Mux4 is disabled to drive the OUTPUT6 of OUTPUT-XBAR<br>1: Respective output of Mux4 is enabled to drive the OUTPUT6 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-110. 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<br>0: Respective output of Mux3 is disabled to drive the OUTPUT6 of OUTPUT-XBAR<br>1: Respective output of Mux3 is enabled to drive the OUTPUT6 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 2 | MUX2 | R/W | 0h | Selects the output of Mux2 to drive OUTPUT6 of OUTPUT-XBAR<br>0: Respective output of Mux2 is disabled to drive the OUTPUT6 of OUTPUT-XBAR<br>1: Respective output of Mux2 is enabled to drive the OUTPUT6 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1 | MUX1 | R/W | 0h | Selects the output of Mux1 to drive OUTPUT6 of OUTPUT-XBAR<br>0: Respective output of Mux1 is disabled to drive the OUTPUT6 of OUTPUT-XBAR<br>1: Respective output of Mux1 is enabled to drive the OUTPUT6 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 0 | MUX0 | R/W | 0h | Selects the output of mux0 to drive OUTPUT6 of OUTPUT-XBAR<br>0: Respective output of Mux0 is disabled to drive the OUTPUT6 of OUTPUT-XBAR<br>1: Respective output of Mux0 is enabled to drive the OUTPUT6 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.6.23 OUTPUT7MUXENABLE Register (Offset = 2Ch) [Reset = 00000000h]

OUTPUT7MUXENABLE is shown in Figure 9-102 and described in Table 9-111.

Return to the Summary Table.

Output X-BAR Mux Enable for Output 7

**Figure 9-102. OUTPUT7MUXENABLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MUX31 | MUX30 | MUX29 | MUX28 | MUX27 | MUX26 | MUX25 | MUX24 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MUX23 | MUX22 | MUX21 | MUX20 | MUX19 | MUX18 | MUX17 | MUX16 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MUX15 | MUX14 | MUX13 | MUX12 | MUX11 | MUX10 | MUX9 | MUX8 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | MUX6 | MUX5 | MUX4 | MUX3 | MUX2 | MUX1 | MUX0 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 9-111. 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<br>0: Respective output of Mux31 is disabled to drive the OUTPUT7 of OUTPUT-XBAR<br>1: Respective output of Mux31 is enabled to drive the OUTPUT7 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 30 | MUX30 | R/W | 0h | Selects the output of Mux30 to drive OUTPUT7 of OUTPUT-XBAR<br>0: Respective output of Mux30 is disabled to drive the OUTPUT7 of OUTPUT-XBAR<br>1: Respective output of Mux30 is enabled to drive the OUTPUT7 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29 | MUX29 | R/W | 0h | Selects the output of Mux29 to drive OUTPUT7 of OUTPUT-XBAR<br>0: Respective output of Mux29 is disabled to drive the OUTPUT7 of OUTPUT-XBAR<br>1: Respective output of Mux29 is enabled to drive the OUTPUT7 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 28 | MUX28 | R/W | 0h | Selects the output of Mux28 to drive OUTPUT7 of OUTPUT-XBAR<br>0: Respective output of Mux28 is disabled to drive the OUTPUT7 of OUTPUT-XBAR<br>1: Respective output of Mux28 is enabled to drive the OUTPUT7 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-111. 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<br>0: Respective output of Mux27 is disabled to drive the OUTPUT7 of OUTPUT-XBAR<br>1: Respective output of Mux27 is enabled to drive the OUTPUT7 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 26 | MUX26 | R/W | 0h | Selects the output of Mux26 to drive OUTPUT7 of OUTPUT-XBAR<br>0: Respective output of Mux26 is disabled to drive the OUTPUT7 of OUTPUT-XBAR<br>1: Respective output of Mux26 is enabled to drive the OUTPUT7 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25 | MUX25 | R/W | 0h | Selects the output of Mux25 to drive OUTPUT7 of OUTPUT-XBAR<br>0: Respective output of Mux25 is disabled to drive the OUTPUT7 of OUTPUT-XBAR<br>1: Respective output of Mux25 is enabled to drive the OUTPUT7 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 24 | MUX24 | R/W | 0h | Selects the output of Mux24 to drive OUTPUT7 of OUTPUT-XBAR<br>0: Respective output of Mux24 is disabled to drive the OUTPUT7 of OUTPUT-XBAR<br>1: Respective output of Mux24 is enabled to drive the OUTPUT7 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23 | MUX23 | R/W | 0h | Selects the output of Mux23 to drive OUTPUT7 of OUTPUT-XBAR<br>0: Respective output of Mux23 is disabled to drive the OUTPUT7 of OUTPUT-XBAR<br>1: Respective output of Mux23 is enabled to drive the OUTPUT7 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 22 | MUX22 | R/W | 0h | Selects the output of Mux22 to drive OUTPUT7 of OUTPUT-XBAR<br>0: Respective output of Mux22 is disabled to drive the OUTPUT7 of OUTPUT-XBAR<br>1: Respective output of Mux22 is enabled to drive the OUTPUT7 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21 | MUX21 | R/W | 0h | Selects the output of Mux21 to drive OUTPUT7 of OUTPUT-XBAR<br>0: Respective output of Mux21 is disabled to drive the OUTPUT7 of OUTPUT-XBAR<br>1: Respective output of Mux21 is enabled to drive the OUTPUT7 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 20 | MUX20 | R/W | 0h | Selects the output of Mux20 to drive OUTPUT7 of OUTPUT-XBAR<br>0: Respective output of Mux20 is disabled to drive the OUTPUT7 of OUTPUT-XBAR<br>1: Respective output of Mux20 is enabled to drive the OUTPUT7 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-111. 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<br>0: Respective output of Mux19 is disabled to drive the OUTPUT7 of OUTPUT-XBAR<br>1: Respective output of Mux19 is enabled to drive the OUTPUT7 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 18 | MUX18 | R/W | 0h | Selects the output of Mux18 to drive OUTPUT7 of OUTPUT-XBAR<br>0: Respective output of Mux18 is disabled to drive the OUTPUT7 of OUTPUT-XBAR<br>1: Respective output of Mux18 is enabled to drive the OUTPUT7 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17 | MUX17 | R/W | 0h | Selects the output of Mux17 to drive OUTPUT7 of OUTPUT-XBAR<br>0: Respective output of Mux17 is disabled to drive the OUTPUT7 of OUTPUT-XBAR<br>1: Respective output of Mux17 is enabled to drive the OUTPUT7 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 16 | MUX16 | R/W | 0h | Selects the output of Mux16 to drive OUTPUT7 of OUTPUT-XBAR<br>0: Respective output of Mux16 is disabled to drive the OUTPUT7 of OUTPUT-XBAR<br>1: Respective output of Mux16 is enabled to drive the OUTPUT7 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15 | MUX15 | R/W | 0h | Selects the output of Mux15 to drive OUTPUT7 of OUTPUT-XBAR<br>0: Respective output of Mux15 is disabled to drive the OUTPUT7 of OUTPUT-XBAR<br>1: Respective output of Mux15 is enabled to drive the OUTPUT7 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 14 | MUX14 | R/W | 0h | Selects the output of Mux14 to drive OUTPUT7 of OUTPUT-XBAR<br>0: Respective output of Mux14 is disabled to drive the OUTPUT7 of OUTPUT-XBAR<br>1: Respective output of Mux14 is enabled to drive the OUTPUT7 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13 | MUX13 | R/W | 0h | Selects the output of Mux13 to drive OUTPUT7 of OUTPUT-XBAR<br>0: Respective output of Mux13 is disabled to drive the OUTPUT7 of OUTPUT-XBAR<br>1: Respective output of Mux13 is enabled to drive the OUTPUT7 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 12 | MUX12 | R/W | 0h | Selects the output of Mux12 to drive OUTPUT7 of OUTPUT-XBAR<br>0: Respective output of Mux12 is disabled to drive the OUTPUT7 of OUTPUT-XBAR<br>1: Respective output of Mux12 is enabled to drive the OUTPUT7 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-111. 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<br>0: Respective output of Mux11 is disabled to drive the OUTPUT7 of OUTPUT-XBAR<br>1: Respective output of Mux11 is enabled to drive the OUTPUT7 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 10 | MUX10 | R/W | 0h | Selects the output of Mux10 to drive OUTPUT7 of OUTPUT-XBAR<br>0: Respective output of Mux10 is disabled to drive the OUTPUT7 of OUTPUT-XBAR<br>1: Respective output of Mux10 is enabled to drive the OUTPUT7 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9 | MUX9 | R/W | 0h | Selects the output of Mux9 to drive OUTPUT7 of OUTPUT-XBAR<br>0: Respective output of Mux9 is disabled to drive the OUTPUT7 of OUTPUT-XBAR<br>1: Respective output of Mux9 is enabled to drive the OUTPUT7 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 8 | MUX8 | R/W | 0h | Selects the output of Mux8 to drive OUTPUT7 of OUTPUT-XBAR<br>0: Respective output of Mux8 is disabled to drive the OUTPUT7 of OUTPUT-XBAR<br>1: Respective output of Mux8 is enabled to drive the OUTPUT7 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7 | MUX7 | R/W | 0h | Selects the output of Mux7 to drive OUTPUT7 of OUTPUT-XBAR<br>0: Respective output of Mux7 is disabled to drive the OUTPUT7 of OUTPUT-XBAR<br>1: Respective output of Mux7 is enabled to drive the OUTPUT7 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 6 | MUX6 | R/W | 0h | Selects the output of Mux6 to drive OUTPUT7 of OUTPUT-XBAR<br>0: Respective output of Mux6 is disabled to drive the OUTPUT7 of OUTPUT-XBAR<br>1: Respective output of Mux6 is enabled to drive the OUTPUT7 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5 | MUX5 | R/W | 0h | Selects the output of Mux5 to drive OUTPUT7 of OUTPUT-XBAR<br>0: Respective output of Mux5 is disabled to drive the OUTPUT7 of OUTPUT-XBAR<br>1: Respective output of Mux5 is enabled to drive the OUTPUT7 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 4 | MUX4 | R/W | 0h | Selects the output of Mux4 to drive OUTPUT7 of OUTPUT-XBAR<br>0: Respective output of Mux4 is disabled to drive the OUTPUT7 of OUTPUT-XBAR<br>1: Respective output of Mux4 is enabled to drive the OUTPUT7 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-111. 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<br>0: Respective output of Mux3 is disabled to drive the OUTPUT7 of OUTPUT-XBAR<br>1: Respective output of Mux3 is enabled to drive the OUTPUT7 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 2 | MUX2 | R/W | 0h | Selects the output of Mux2 to drive OUTPUT7 of OUTPUT-XBAR<br>0: Respective output of Mux2 is disabled to drive the OUTPUT7 of OUTPUT-XBAR<br>1: Respective output of Mux2 is enabled to drive the OUTPUT7 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1 | MUX1 | R/W | 0h | Selects the output of Mux1 to drive OUTPUT7 of OUTPUT-XBAR<br>0: Respective output of Mux1 is disabled to drive the OUTPUT7 of OUTPUT-XBAR<br>1: Respective output of Mux1 is enabled to drive the OUTPUT7 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 0 | MUX0 | R/W | 0h | Selects the output of mux0 to drive OUTPUT7 of OUTPUT-XBAR<br>0: Respective output of Mux0 is disabled to drive the OUTPUT7 of OUTPUT-XBAR<br>1: Respective output of Mux0 is enabled to drive the OUTPUT7 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.6.24 OUTPUT8MUXENABLE Register (Offset = 2Eh) [Reset = 00000000h]

OUTPUT8MUXENABLE is shown in Figure 9-103 and described in Table 9-112.

Return to the Summary Table.

Output X-BAR Mux Enable for Output 8

**Figure 9-103. OUTPUT8MUXENABLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MUX31 | MUX30 | MUX29 | MUX28 | MUX27 | MUX26 | MUX25 | MUX24 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MUX23 | MUX22 | MUX21 | MUX20 | MUX19 | MUX18 | MUX17 | MUX16 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MUX15 | MUX14 | MUX13 | MUX12 | MUX11 | MUX10 | MUX9 | MUX8 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | MUX6 | MUX5 | MUX4 | MUX3 | MUX2 | MUX1 | MUX0 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 9-112. 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<br>0: Respective output of Mux31 is disabled to drive the OUTPUT8 of OUTPUT-XBAR<br>1: Respective output of Mux31 is enabled to drive the OUTPUT8 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 30 | MUX30 | R/W | 0h | Selects the output of Mux30 to drive OUTPUT8 of OUTPUT-XBAR<br>0: Respective output of Mux30 is disabled to drive the OUTPUT8 of OUTPUT-XBAR<br>1: Respective output of Mux30 is enabled to drive the OUTPUT8 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 29 | MUX29 | R/W | 0h | Selects the output of Mux29 to drive OUTPUT8 of OUTPUT-XBAR<br>0: Respective output of Mux29 is disabled to drive the OUTPUT8 of OUTPUT-XBAR<br>1: Respective output of Mux29 is enabled to drive the OUTPUT8 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 28 | MUX28 | R/W | 0h | Selects the output of Mux28 to drive OUTPUT8 of OUTPUT-XBAR<br>0: Respective output of Mux28 is disabled to drive the OUTPUT8 of OUTPUT-XBAR<br>1: Respective output of Mux28 is enabled to drive the OUTPUT8 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-112. 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<br>0: Respective output of Mux27 is disabled to drive the OUTPUT8 of OUTPUT-XBAR<br>1: Respective output of Mux27 is enabled to drive the OUTPUT8 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 26 | MUX26 | R/W | 0h | Selects the output of Mux26 to drive OUTPUT8 of OUTPUT-XBAR<br>0: Respective output of Mux26 is disabled to drive the OUTPUT8 of OUTPUT-XBAR<br>1: Respective output of Mux26 is enabled to drive the OUTPUT8 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 25 | MUX25 | R/W | 0h | Selects the output of Mux25 to drive OUTPUT8 of OUTPUT-XBAR<br>0: Respective output of Mux25 is disabled to drive the OUTPUT8 of OUTPUT-XBAR<br>1: Respective output of Mux25 is enabled to drive the OUTPUT8 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 24 | MUX24 | R/W | 0h | Selects the output of Mux24 to drive OUTPUT8 of OUTPUT-XBAR<br>0: Respective output of Mux24 is disabled to drive the OUTPUT8 of OUTPUT-XBAR<br>1: Respective output of Mux24 is enabled to drive the OUTPUT8 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 23 | MUX23 | R/W | 0h | Selects the output of Mux23 to drive OUTPUT8 of OUTPUT-XBAR<br>0: Respective output of Mux23 is disabled to drive the OUTPUT8 of OUTPUT-XBAR<br>1: Respective output of Mux23 is enabled to drive the OUTPUT8 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 22 | MUX22 | R/W | 0h | Selects the output of Mux22 to drive OUTPUT8 of OUTPUT-XBAR<br>0: Respective output of Mux22 is disabled to drive the OUTPUT8 of OUTPUT-XBAR<br>1: Respective output of Mux22 is enabled to drive the OUTPUT8 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 21 | MUX21 | R/W | 0h | Selects the output of Mux21 to drive OUTPUT8 of OUTPUT-XBAR<br>0: Respective output of Mux21 is disabled to drive the OUTPUT8 of OUTPUT-XBAR<br>1: Respective output of Mux21 is enabled to drive the OUTPUT8 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 20 | MUX20 | R/W | 0h | Selects the output of Mux20 to drive OUTPUT8 of OUTPUT-XBAR<br>0: Respective output of Mux20 is disabled to drive the OUTPUT8 of OUTPUT-XBAR<br>1: Respective output of Mux20 is enabled to drive the OUTPUT8 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-112. 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<br>0: Respective output of Mux19 is disabled to drive the OUTPUT8 of OUTPUT-XBAR<br>1: Respective output of Mux19 is enabled to drive the OUTPUT8 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 18 | MUX18 | R/W | 0h | Selects the output of Mux18 to drive OUTPUT8 of OUTPUT-XBAR<br>0: Respective output of Mux18 is disabled to drive the OUTPUT8 of OUTPUT-XBAR<br>1: Respective output of Mux18 is enabled to drive the OUTPUT8 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 17 | MUX17 | R/W | 0h | Selects the output of Mux17 to drive OUTPUT8 of OUTPUT-XBAR<br>0: Respective output of Mux17 is disabled to drive the OUTPUT8 of OUTPUT-XBAR<br>1: Respective output of Mux17 is enabled to drive the OUTPUT8 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 16 | MUX16 | R/W | 0h | Selects the output of Mux16 to drive OUTPUT8 of OUTPUT-XBAR<br>0: Respective output of Mux16 is disabled to drive the OUTPUT8 of OUTPUT-XBAR<br>1: Respective output of Mux16 is enabled to drive the OUTPUT8 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 15 | MUX15 | R/W | 0h | Selects the output of Mux15 to drive OUTPUT8 of OUTPUT-XBAR<br>0: Respective output of Mux15 is disabled to drive the OUTPUT8 of OUTPUT-XBAR<br>1: Respective output of Mux15 is enabled to drive the OUTPUT8 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 14 | MUX14 | R/W | 0h | Selects the output of Mux14 to drive OUTPUT8 of OUTPUT-XBAR<br>0: Respective output of Mux14 is disabled to drive the OUTPUT8 of OUTPUT-XBAR<br>1: Respective output of Mux14 is enabled to drive the OUTPUT8 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 13 | MUX13 | R/W | 0h | Selects the output of Mux13 to drive OUTPUT8 of OUTPUT-XBAR<br>0: Respective output of Mux13 is disabled to drive the OUTPUT8 of OUTPUT-XBAR<br>1: Respective output of Mux13 is enabled to drive the OUTPUT8 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 12 | MUX12 | R/W | 0h | Selects the output of Mux12 to drive OUTPUT8 of OUTPUT-XBAR<br>0: Respective output of Mux12 is disabled to drive the OUTPUT8 of OUTPUT-XBAR<br>1: Respective output of Mux12 is enabled to drive the OUTPUT8 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-112. 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<br>0: Respective output of Mux11 is disabled to drive the OUTPUT8 of OUTPUT-XBAR<br>1: Respective output of Mux11 is enabled to drive the OUTPUT8 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 10 | MUX10 | R/W | 0h | Selects the output of Mux10 to drive OUTPUT8 of OUTPUT-XBAR<br>0: Respective output of Mux10 is disabled to drive the OUTPUT8 of OUTPUT-XBAR<br>1: Respective output of Mux10 is enabled to drive the OUTPUT8 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 9 | MUX9 | R/W | 0h | Selects the output of Mux9 to drive OUTPUT8 of OUTPUT-XBAR<br>0: Respective output of Mux9 is disabled to drive the OUTPUT8 of OUTPUT-XBAR<br>1: Respective output of Mux9 is enabled to drive the OUTPUT8 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 8 | MUX8 | R/W | 0h | Selects the output of Mux8 to drive OUTPUT8 of OUTPUT-XBAR<br>0: Respective output of Mux8 is disabled to drive the OUTPUT8 of OUTPUT-XBAR<br>1: Respective output of Mux8 is enabled to drive the OUTPUT8 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 7 | MUX7 | R/W | 0h | Selects the output of Mux7 to drive OUTPUT8 of OUTPUT-XBAR<br>0: Respective output of Mux7 is disabled to drive the OUTPUT8 of OUTPUT-XBAR<br>1: Respective output of Mux7 is enabled to drive the OUTPUT8 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 6 | MUX6 | R/W | 0h | Selects the output of Mux6 to drive OUTPUT8 of OUTPUT-XBAR<br>0: Respective output of Mux6 is disabled to drive the OUTPUT8 of OUTPUT-XBAR<br>1: Respective output of Mux6 is enabled to drive the OUTPUT8 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5 | MUX5 | R/W | 0h | Selects the output of Mux5 to drive OUTPUT8 of OUTPUT-XBAR<br>0: Respective output of Mux5 is disabled to drive the OUTPUT8 of OUTPUT-XBAR<br>1: Respective output of Mux5 is enabled to drive the OUTPUT8 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 4 | MUX4 | R/W | 0h | Selects the output of Mux4 to drive OUTPUT8 of OUTPUT-XBAR<br>0: Respective output of Mux4 is disabled to drive the OUTPUT8 of OUTPUT-XBAR<br>1: Respective output of Mux4 is enabled to drive the OUTPUT8 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-112. 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<br>0: Respective output of Mux3 is disabled to drive the OUTPUT8 of OUTPUT-XBAR<br>1: Respective output of Mux3 is enabled to drive the OUTPUT8 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 2 | MUX2 | R/W | 0h | Selects the output of Mux2 to drive OUTPUT8 of OUTPUT-XBAR<br>0: Respective output of Mux2 is disabled to drive the OUTPUT8 of OUTPUT-XBAR<br>1: Respective output of Mux2 is enabled to drive the OUTPUT8 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1 | MUX1 | R/W | 0h | Selects the output of Mux1 to drive OUTPUT8 of OUTPUT-XBAR<br>0: Respective output of Mux1 is disabled to drive the OUTPUT8 of OUTPUT-XBAR<br>1: Respective output of Mux1 is enabled to drive the OUTPUT8 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 0 | MUX0 | R/W | 0h | Selects the output of mux0 to drive OUTPUT8 of OUTPUT-XBAR<br>0: Respective output of Mux0 is disabled to drive the OUTPUT8 of OUTPUT-XBAR<br>1: Respective output of Mux0 is enabled to drive the OUTPUT8 of OUTPUT-XBAR<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.6.25 OUTPUTLATCH Register (Offset = 30h) [Reset = 00000000h]

OUTPUTLATCH is shown in Figure 9-104 and described in Table 9-113.

Return to the Summary Table.

Output X-BAR Output Latch

**Figure 9-104. 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 9-113. 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.<br>0: Respective output has not been triggered<br>1: Respective output is triggered<br>Refer to the Output X-BAR section of this chapter for more details.<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 6 | OUTPUT7 | R | 0h | Records the OUTPUT7 of OUTPUT-XBAR.<br>0: Respective output has not been triggered<br>1: Respective output is triggered<br>Refer to the Output X-BAR section of this chapter for more details.<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 5 | OUTPUT6 | R | 0h | Records the OUTPUT6 of OUTPUT-XBAR.<br>0: Respective output has not been triggered<br>1: Respective output is triggered<br>Refer to the Output X-BAR section of this chapter for more details.<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 4 | OUTPUT5 | R | 0h | Records the OUTPUT5 of OUTPUT-XBAR.<br>0: Respective output has not been triggered<br>1: Respective output is triggered<br>Refer to the Output X-BAR section of this chapter for more details.<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-113. OUTPUTLATCH Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3 | OUTPUT4 | R | 0h | Records the OUTPUT4 of OUTPUT-XBAR.<br>0: Respective output has not been triggered<br>1: Respective output is triggered<br>Refer to the Output X-BAR section of this chapter for more details.<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 2 | OUTPUT3 | R | 0h | Records the OUTPUT3 of OUTPUT-XBAR.<br>0: Respective output has not been triggered<br>1: Respective output is triggered<br>Refer to the Output X-BAR section of this chapter for more details.<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 1 | OUTPUT2 | R | 0h | Records the OUTPUT2 of OUTPUT-XBAR.<br>0: Respective output has not been triggered<br>1: Respective output is triggered<br>Refer to the Output X-BAR section of this chapter for more details.<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 0 | OUTPUT1 | R | 0h | Records the OUTPUT1 of OUTPUT-XBAR.<br>0: Respective output has not been triggered<br>1: Respective output is triggered<br>Refer to the Output X-BAR section of this chapter for more details.<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.6.26 OUTPUTLATCHCLR Register (Offset = 32h) [Reset = 00000000h]

OUTPUTLATCHCLR is shown in Figure 9-105 and described in Table 9-114.

Return to the Summary Table.

Output X-BAR Output Latch Clear

**Figure 9-105. OUTPUTLATCHCLR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OUTPUT8 | OUTPUT7 | OUTPUT6 | OUTPUT5 | OUTPUT4 | OUTPUT3 | OUTPUT2 | OUTPUT1 |
| R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h |

**Table 9-114. 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<br>Writing 1 clears the corresponding output latch bit in the OUTPUTLATCH register<br>Write of 0 has no effect<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 6 | OUTPUT7 | R-0/W1S | 0h | Clears the Output-Latch for OUTPUT7 of OUTPUT-XBAR<br>Writing 1 clears the corresponding output latch bit in the OUTPUTLATCH register<br>Write of 0 has no effect<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5 | OUTPUT6 | R-0/W1S | 0h | Clears the Output-Latch for OUTPUT6 of OUTPUT-XBAR<br>Writing 1 clears the corresponding output latch bit in the OUTPUTLATCH register<br>Write of 0 has no effect<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 4 | OUTPUT5 | R-0/W1S | 0h | Clears the Output-Latch for OUTPUT5 of OUTPUT-XBAR<br>Writing 1 clears the corresponding output latch bit in the OUTPUTLATCH register<br>Write of 0 has no effect<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 3 | OUTPUT4 | R-0/W1S | 0h | Clears the Output-Latch for OUTPUT4 of OUTPUT-XBAR<br>Writing 1 clears the corresponding output latch bit in the OUTPUTLATCH register<br>Write of 0 has no effect<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-114. 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<br>Writing 1 clears the corresponding output latch bit in the OUTPUTLATCH register<br>Write of 0 has no effect<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1 | OUTPUT2 | R-0/W1S | 0h | Clears the Output-Latch for OUTPUT2 of OUTPUT-XBAR<br>Writing 1 clears the corresponding output latch bit in the OUTPUTLATCH register<br>Write of 0 has no effect<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 0 | OUTPUT1 | R-0/W1S | 0h | Clears the Output-Latch for OUTPUT1 of OUTPUT-XBAR<br>Writing 1 clears the corresponding output latch bit in the OUTPUTLATCH register<br>Write of 0 has no effect<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.6.27 OUTPUTLATCHFRC Register (Offset = 34h) [Reset = 00000000h]

OUTPUTLATCHFRC is shown in Figure 9-106 and described in Table 9-115.

Return to the Summary Table.

Output X-BAR Output Latch Clear

**Figure 9-106. OUTPUTLATCHFRC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OUTPUT8 | OUTPUT7 | OUTPUT6 | OUTPUT5 | OUTPUT4 | OUTPUT3 | OUTPUT2 | OUTPUT1 |
| R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h |

**Table 9-115. 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<br>Writing 1 sets the corresponding output latch bit in the OUTPUTLATCH register<br>Write of 0 has no effect<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 6 | OUTPUT7 | R-0/W1S | 0h | Sets the Output-Latch for OUTPUT7 of OUTPUT-XBAR<br>Writing 1 sets the corresponding output latch bit in the OUTPUTLATCH register<br>Write of 0 has no effect<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5 | OUTPUT6 | R-0/W1S | 0h | Sets the Output-Latch for OUTPUT6 of OUTPUT-XBAR<br>Writing 1 sets the corresponding output latch bit in the OUTPUTLATCH register<br>Write of 0 has no effect<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 4 | OUTPUT5 | R-0/W1S | 0h | Sets the Output-Latch for OUTPUT5 of OUTPUT-XBAR<br>Writing 1 sets the corresponding output latch bit in the OUTPUTLATCH register<br>Write of 0 has no effect<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 3 | OUTPUT4 | R-0/W1S | 0h | Sets the Output-Latch for OUTPUT4 of OUTPUT-XBAR<br>Writing 1 sets the corresponding output latch bit in the OUTPUTLATCH register<br>Write of 0 has no effect<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-115. 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<br>Writing 1 sets the corresponding output latch bit in the OUTPUTLATCH register<br>Write of 0 has no effect<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 1 | OUTPUT2 | R-0/W1S | 0h | Sets the Output-Latch for OUTPUT2 of OUTPUT-XBAR<br>Writing 1 sets the corresponding output latch bit in the OUTPUTLATCH register<br>Write of 0 has no effect<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 0 | OUTPUT1 | R-0/W1S | 0h | Sets the Output-Latch for OUTPUT1 of OUTPUT-XBAR<br>Writing 1 sets the corresponding output latch bit in the OUTPUTLATCH register<br>Write of 0 has no effect<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.6.28 OUTPUTLATCHENABLE Register (Offset = 36h) [Reset = 00000000h]

OUTPUTLATCHENABLE is shown in Figure 9-107 and described in Table 9-116.

Return to the Summary Table.

Output X-BAR Output Latch Enable

**Figure 9-107. 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 9-116. 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<br>0: Output Latch is not selected to driven the respective output<br>1: Output Latch is selected to drive the respective output<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 6 | OUTPUT7 | R/W | 0h | Selects the output latch to drive OUTPUT7 for OUTPUT-XBAR<br>0: Output Latch is not selected to driven the respective output<br>1: Output Latch is selected to drive the respective output<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5 | OUTPUT6 | R/W | 0h | Selects the output latch to drive OUTPUT6 for OUTPUT-XBAR<br>0: Output Latch is not selected to driven the respective output<br>1: Output Latch is selected to drive the respective output<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 4 | OUTPUT5 | R/W | 0h | Selects the output latch to drive OUTPUT5 for OUTPUT-XBAR<br>0: Output Latch is not selected to driven the respective output<br>1: Output Latch is selected to drive the respective output<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 3 | OUTPUT4 | R/W | 0h | Selects the output latch to drive OUTPUT4 for OUTPUT-XBAR<br>0: Output Latch is not selected to driven the respective output<br>1: Output Latch is selected to drive the respective output<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 2 | OUTPUT3 | R/W | 0h | Selects the output latch to drive OUTPUT3 for OUTPUT-XBAR<br>0: Output Latch is not selected to driven the respective output<br>1: Output Latch is selected to drive the respective output<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-116. 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<br>0: Output Latch is not selected to driven the respective output<br>1: Output Latch is selected to drive the respective output<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 0 | OUTPUT1 | R/W | 0h | Selects the output latch to drive OUTPUT1 for OUTPUT-XBAR<br>0: Output Latch is not selected to driven the respective output<br>1: Output Latch is selected to drive the respective output<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.6.29 OUTPUTINV Register (Offset = 38h) [Reset = 00000000h]

OUTPUTINV is shown in Figure 9-108 and described in Table 9-117.

Return to the Summary Table.

Output X-BAR Output Inversion

**Figure 9-108. 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 9-117. 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<br>0: drives active high output<br>1: drives active-low output<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 6 | OUTPUT7 | R/W | 0h | Selects polarity for OUTPUT7 of OUTPUT-XBAR<br>0: drives active high output<br>1: drives active-low output<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 5 | OUTPUT6 | R/W | 0h | Selects polarity for OUTPUT6 of OUTPUT-XBAR<br>0: drives active high output<br>1: drives active-low output<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 4 | OUTPUT5 | R/W | 0h | Selects polarity for OUTPUT5 of OUTPUT-XBAR<br>0: drives active high output<br>1: drives active-low output<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 3 | OUTPUT4 | R/W | 0h | Selects polarity for OUTPUT4 of OUTPUT-XBAR<br>0: drives active high output<br>1: drives active-low output<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 2 | OUTPUT3 | R/W | 0h | Selects polarity for OUTPUT3 of OUTPUT-XBAR<br>0: drives active high output<br>1: drives active-low output<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-117. OUTPUTINV Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1 | OUTPUT2 | R/W | 0h | Selects polarity for OUTPUT2 of OUTPUT-XBAR<br>0: drives active high output<br>1: drives active-low output<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |
| 0 | OUTPUT1 | R/W | 0h | Selects polarity for OUTPUT1 of OUTPUT-XBAR<br>0: drives active high output<br>1: drives active-low output<br>Refer to the Output X-BAR section of this chapter for more details.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.6.30 OUTPUTLOCK Register (Offset = 3Eh) [Reset = 00000000h]

OUTPUTLOCK is shown in Figure 9-109 and described in Table 9-118.

Return to the Summary Table.

Output X-BAR Configuration Lock register

**Figure 9-109. 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/WSonce-0h |

**Table 9-118. 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<br>Reset type: CPU1.SYSRSn |
| 15-1 | RESERVED | R-0 | 0h | Reserved |
| 0 | LOCK | R/WSonce | 0h | Locks the configuration for OUTPUT-XBAR. Once the configuration is locked, writes to the below registers for OUTPUT-XBAR is blocked.<br>Registers Affected by the LOCK mechanism:<br>OUTPUT-XBAROUTyMUX0TO15CFG<br>OUTPUT-XBAROUTyMUX16TO31CFG<br>OUTPUT-XBAROUTyMUXENABLE<br>OUTPUT-XBAROUTLATENABLE<br>OUTPUT-XBAROUTINV<br>0: Writes to the above registers are allowed<br>1: Writes to the above registers are blocked<br>Note:<br>[1] LOCK mechanism only apples to writes. Reads are never blocked.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 9.3.7 Register to Driverlib Function Mapping

#### 9.3.7.1 INPUTXBAR Registers to Driverlib Functions

**Table 9-119. INPUTXBAR Registers to Driverlib Functions**

| File | Driverlib Function |
|---|---|
| **INPUT1SELECT** | |
| xbar.h | XBAR_setInputPin |
| **INPUT2SELECT** | |
| - | See INPUT1SELECT |
| **INPUT3SELECT** | |
| - | See INPUT1SELECT |
| **INPUT4SELECT** | |
| - | See INPUT1SELECT |
| **INPUT5SELECT** | |
| - | See INPUT1SELECT |
| **INPUT6SELECT** | |
| - | See INPUT1SELECT |
| **INPUT7SELECT** | |
| - | See INPUT1SELECT |
| **INPUT8SELECT** | |
| - | See INPUT1SELECT |
| **INPUT9SELECT** | |
| - | See INPUT1SELECT |
| **INPUT10SELECT** | |
| - | See INPUT1SELECT |
| **INPUT11SELECT** | |
| - | See INPUT1SELECT |
| **INPUT12SELECT** | |
| - | See INPUT1SELECT |
| **INPUT13SELECT** | |
| - | See INPUT1SELECT |
| **INPUT14SELECT** | |
| - | See INPUT1SELECT |
| **INPUTSELECTLOCK** | |
| xbar.h | XBAR_lockInput |

#### 9.3.7.2 XBAR Registers to Driverlib Functions

**Table 9-120. XBAR Registers to Driverlib Functions**

| File | Driverlib Function |
|---|---|
| **FLG1** | |
| xbar.c | XBAR_getInputFlagStatus |
| **FLG2** | |
| xbar.c | XBAR_getInputFlagStatus |
| **FLG3** | |
| xbar.c | XBAR_getInputFlagStatus |
| **CLR1** | |
| xbar.c | XBAR_clearInputFlag |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-120. XBAR Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| **CLR2** | |
| xbar.c | XBAR_clearInputFlag |
| **CLR3** | |
| xbar.c | XBAR_clearInputFlag |

#### 9.3.7.3 EPWMXBAR Registers to Driverlib Functions

**Table 9-121. EPWMXBAR Registers to Driverlib Functions**

| File | Driverlib Function |
|---|---|
| **TRIP4MUX0TO15CFG** | |
| xbar.c | XBAR_setEPWMMuxConfig |
| **TRIP4MUX16TO31CFG** | |
| xbar.c | XBAR_setEPWMMuxConfig |
| **TRIP5MUX0TO15CFG** | |
| - | See TRIP4MUX0TO15CFG |
| **TRIP5MUX16TO31CFG** | |
| - | See TRIP4MUX0TO15CFG |
| **TRIP7MUX0TO15CFG** | |
| - | See TRIP4MUX0TO15CFG |
| **TRIP7MUX16TO31CFG** | |
| - | See TRIP4MUX0TO15CFG |
| **TRIP8MUX0TO15CFG** | |
| - | See TRIP4MUX0TO15CFG |
| **TRIP8MUX16TO31CFG** | |
| - | See TRIP4MUX0TO15CFG |
| **TRIP9MUX0TO15CFG** | |
| - | See TRIP4MUX0TO15CFG |
| **TRIP9MUX16TO31CFG** | |
| - | See TRIP4MUX0TO15CFG |
| **TRIP10MUX0TO15CFG** | |
| - | See TRIP4MUX0TO15CFG |
| **TRIP10MUX16TO31CFG** | |
| - | See TRIP4MUX0TO15CFG |
| **TRIP11MUX0TO15CFG** | |
| - | See TRIP4MUX0TO15CFG |
| **TRIP11MUX16TO31CFG** | |
| - | See TRIP4MUX0TO15CFG |
| **TRIP12MUX0TO15CFG** | |
| - | See TRIP4MUX0TO15CFG |
| **TRIP12MUX16TO31CFG** | |
| - | See TRIP4MUX0TO15CFG |
| **TRIP4MUXENABLE** | |
| xbar.h | XBAR_enableEPWMMux |
| xbar.h | XBAR_disableEPWMMux |
| **TRIP5MUXENABLE** | |
| - | See TRIP4MUXENABLE |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-121. EPWMXBAR Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| **TRIP7MUXENABLE** | |
| - | See TRIP4MUXENABLE |
| **TRIP8MUXENABLE** | |
| - | See TRIP4MUXENABLE |
| **TRIP9MUXENABLE** | |
| - | See TRIP4MUXENABLE |
| **TRIP10MUXENABLE** | |
| - | See TRIP4MUXENABLE |
| **TRIP11MUXENABLE** | |
| - | See TRIP4MUXENABLE |
| **TRIP12MUXENABLE** | |
| - | See TRIP4MUXENABLE |
| **TRIPOUTINV** | |
| xbar.h | XBAR_invertEPWMSignal |
| **TRIPLOCK** | |
| xbar.h | XBAR_lockEPWM |

### 9.3.7.4 CLBXBAR Registers to Driverlib Functions

**Table 9-122. CLBXBAR Registers to Driverlib Functions**

| File | Driverlib Function |
|---|---|
| **AUXSIG0MUX0TO15CFG** | |
| xbar.c | XBAR_setCLBMuxConfig |
| **AUXSIG0MUX16TO31CFG** | |
| xbar.c | XBAR_setCLBMuxConfig |
| **AUXSIG1MUX0TO15CFG** | |
| - | See AUXSIG0MUX0TO15CFG |
| **AUXSIG1MUX16TO31CFG** | |
| - | See AUXSIG0MUX0TO15CFG |
| **AUXSIG2MUX0TO15CFG** | |
| - | See AUXSIG0MUX0TO15CFG |
| **AUXSIG2MUX16TO31CFG** | |
| - | See AUXSIG0MUX0TO15CFG |
| **AUXSIG3MUX0TO15CFG** | |
| - | See AUXSIG0MUX0TO15CFG |
| **AUXSIG3MUX16TO31CFG** | |
| - | See AUXSIG0MUX0TO15CFG |
| **AUXSIG4MUX0TO15CFG** | |
| - | See AUXSIG0MUX0TO15CFG |
| **AUXSIG4MUX16TO31CFG** | |
| - | See AUXSIG0MUX0TO15CFG |
| **AUXSIG5MUX0TO15CFG** | |
| - | See AUXSIG0MUX0TO15CFG |
| **AUXSIG5MUX16TO31CFG** | |
| - | See AUXSIG0MUX0TO15CFG |
| **AUXSIG6MUX0TO15CFG** | |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-122. CLBXBAR Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| - | See AUXSIG0MUX0TO15CFG |
| **AUXSIG6MUX16TO31CFG** | |
| - | See AUXSIG0MUX0TO15CFG |
| **AUXSIG7MUX0TO15CFG** | |
| - | See AUXSIG0MUX0TO15CFG |
| **AUXSIG7MUX16TO31CFG** | |
| - | See AUXSIG0MUX0TO15CFG |
| **AUXSIG0MUXENABLE** | |
| xbar.h | XBAR_enableCLBMux |
| xbar.h | XBAR_disableCLBMux |
| **AUXSIG1MUXENABLE** | |
| - | See AUXSIG0MUXENABLE |
| **AUXSIG2MUXENABLE** | |
| - | See AUXSIG0MUXENABLE |
| **AUXSIG3MUXENABLE** | |
| - | See AUXSIG0MUXENABLE |
| **AUXSIG4MUXENABLE** | |
| - | See AUXSIG0MUXENABLE |
| **AUXSIG5MUXENABLE** | |
| - | See AUXSIG0MUXENABLE |
| **AUXSIG6MUXENABLE** | |
| - | See AUXSIG0MUXENABLE |
| **AUXSIG7MUXENABLE** | |
| - | See AUXSIG0MUXENABLE |
| **AUXSIGOUTINV** | |
| xbar.h | XBAR_invertCLBSignal |
| **AUXSIGLOCK** | |
| - | |

### 9.3.7.5 OUTPUTXBAR Registers to Driverlib Functions

**Table 9-123. OUTPUTXBAR Registers to Driverlib Functions**

| File | Driverlib Function |
|---|---|
| **OUTPUT1MUX0TO15CFG** | |
| xbar.c | XBAR_setOutputMuxConfig |
| **OUTPUT1MUX16TO31CFG** | |
| xbar.c | XBAR_setOutputMuxConfig |
| **OUTPUT2MUX0TO15CFG** | |
| - | See OUTPUT1MUX0TO15CFG |
| **OUTPUT2MUX16TO31CFG** | |
| - | See OUTPUT1MUX0TO15CFG |
| **OUTPUT3MUX0TO15CFG** | |
| - | See OUTPUT1MUX0TO15CFG |
| **OUTPUT3MUX16TO31CFG** | |
| - | See OUTPUT1MUX0TO15CFG |
| **OUTPUT4MUX0TO15CFG** | |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-123. OUTPUTXBAR Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| - | See OUTPUT1MUX0TO15CFG |
| **OUTPUT4MUX16TO31CFG** | |
| - | See OUTPUT1MUX0TO15CFG |
| **OUTPUT5MUX0TO15CFG** | |
| - | See OUTPUT1MUX0TO15CFG |
| **OUTPUT5MUX16TO31CFG** | |
| - | See OUTPUT1MUX0TO15CFG |
| **OUTPUT6MUX0TO15CFG** | |
| - | See OUTPUT1MUX0TO15CFG |
| **OUTPUT6MUX16TO31CFG** | |
| - | See OUTPUT1MUX0TO15CFG |
| **OUTPUT7MUX0TO15CFG** | |
| - | See OUTPUT1MUX0TO15CFG |
| **OUTPUT7MUX16TO31CFG** | |
| - | See OUTPUT1MUX0TO15CFG |
| **OUTPUT8MUX0TO15CFG** | |
| - | See OUTPUT1MUX0TO15CFG |
| **OUTPUT8MUX16TO31CFG** | |
| - | See OUTPUT1MUX0TO15CFG |
| **OUTPUT1MUXENABLE** | |
| xbar.h | XBAR_enableOutputMux |
| xbar.h | XBAR_disableOutputMux |
| **OUTPUT2MUXENABLE** | |
| - | See OUTPUT1MUXENABLE |
| **OUTPUT3MUXENABLE** | |
| - | See OUTPUT1MUXENABLE |
| **OUTPUT4MUXENABLE** | |
| - | See OUTPUT1MUXENABLE |
| **OUTPUT5MUXENABLE** | |
| - | See OUTPUT1MUXENABLE |
| **OUTPUT6MUXENABLE** | |
| - | See OUTPUT1MUXENABLE |
| **OUTPUT7MUXENABLE** | |
| - | See OUTPUT1MUXENABLE |
| **OUTPUT8MUXENABLE** | |
| - | See OUTPUT1MUXENABLE |
| **OUTPUTLATCH** | |
| xbar.h | XBAR_setOutputLatchMode |
| xbar.h | XBAR_getOutputLatchStatus |
| xbar.h | XBAR_clearOutputLatch |
| xbar.h | XBAR_forceOutputLatch |
| **OUTPUTLATCHCLR** | |
| xbar.h | XBAR_clearOutputLatch |
| **OUTPUTLATCHFRC** | |
| xbar.h | XBAR_forceOutputLatch |



Copyright © 2024 Texas Instruments Incorporated

**Table 9-123. OUTPUTXBAR Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| **OUTPUTLATCHENABLE** | |
| xbar.h | XBAR_setOutputLatchMode |
| **OUTPUTINV** | |
| xbar.h | XBAR_invertOutputSignal |
| **OUTPUTLOCK** | |
| xbar.h | XBAR_lockOutput |



Copyright © 2024 Texas Instruments Incorporated

*Chapter 10*

# *Analog Subsystem*

The analog subsystem module is described in this chapter.


**10.1 Introduction**.....1566
**10.2 Optimizing Power-Up Time**.....1570
**10.3 Analog Subsystem Registers**.....1571





Copyright © 2024 Texas Instruments Incorporated

## 10.1 Introduction

The analog modules on this device include the Analog-to-Digital Converter (ADC), Temperature Sensor, Buffered Digital-to-Analog Converter (DAC), and Comparator Subsystem (CMPSS).

### *10.1.1 Features*

The analog subsystem has the following features:

- Flexible voltage references
  - The ADCs are referenced to VREFHIx and VREFLOx pins
    - VREFHIx pin voltage must be driven in externally
- The buffered DACs are referenced to VREFHIx and VSSA
  - Alternately, these DACs can be referenced to the VDAC pin and VSSA
- The comparator DACs are referenced to VDDA and VSSA
  - Alternately, these DACs can be referenced to the VDAC pin and VSSA
- Flexible pin usage
  - Buffered DAC and comparator subsystem functions multiplexed with ADC inputs
- Internal connection to VREFLO 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 reference pins, VREFHIA to VREFHID and VREFLOA to VREFLOD, can be used to externally supply the reference to the ADC. VREFHIA can also be used to supply the reference voltage to DAC A and DAC B and VREFHIB can be used to supply the reference to DAC C.

The analog module input and outputs are all ADC inputs by default. The pins that connect to CMPSS inputs can be used for the CMPSS without further action and without preventing use as an ADC input simultaneously. DAC outputs must be enabled; this prevents the channel from simultaneously being used as an ADC input (but the ADC can be used to sample the DAC output voltage, if desired).

The VDAC reference pin can be used to set an alternate range for DAC A, DAC B, and DAC C and for the DACs inside the CMPSS modules (the CMPSS DACs are referenced to VDDA and VSSA by default). Using this pin as a reference prevents the channel from being used as an ADC input (but the ADC can be used to sample the VDAC voltage, if desired). The choice of reference is configurable per-module for each CMPSS or buffered DAC, and the selection is made using the module's configuration registers.

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.
- For buffered DAC modules, VSSA is the low reference whether VREFHIx or VDAC is selected as the high reference.
- For CMPSS modules, VSSA is the low reference whether VDAC or VDDA is selected as the high reference.



Copyright © 2024 Texas Instruments Incorporated

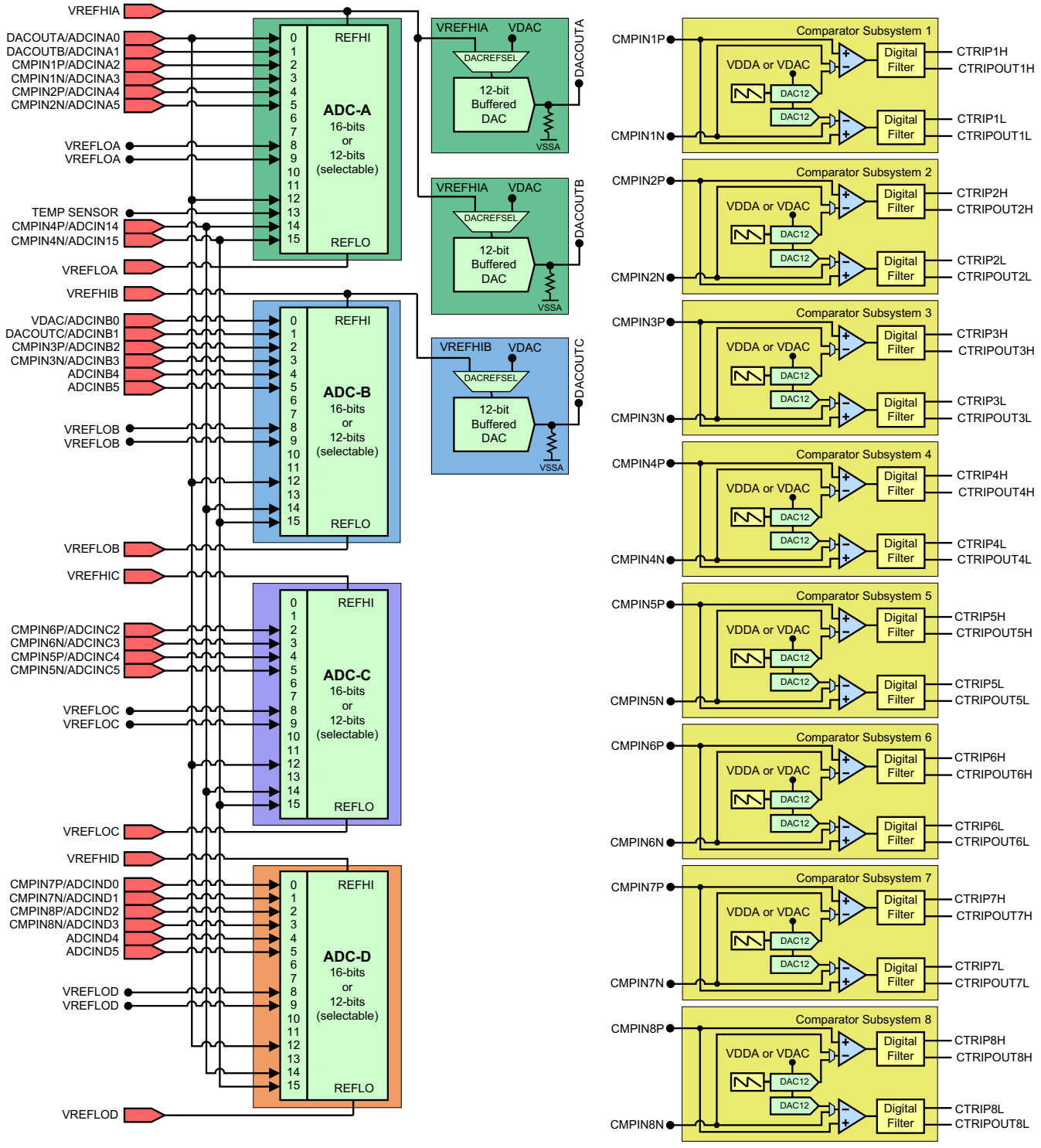



**Figure 10-1. Analog Subsystem Block Diagram (337-Ball ZWT)**


Copyright © 2024 Texas Instruments Incorporated

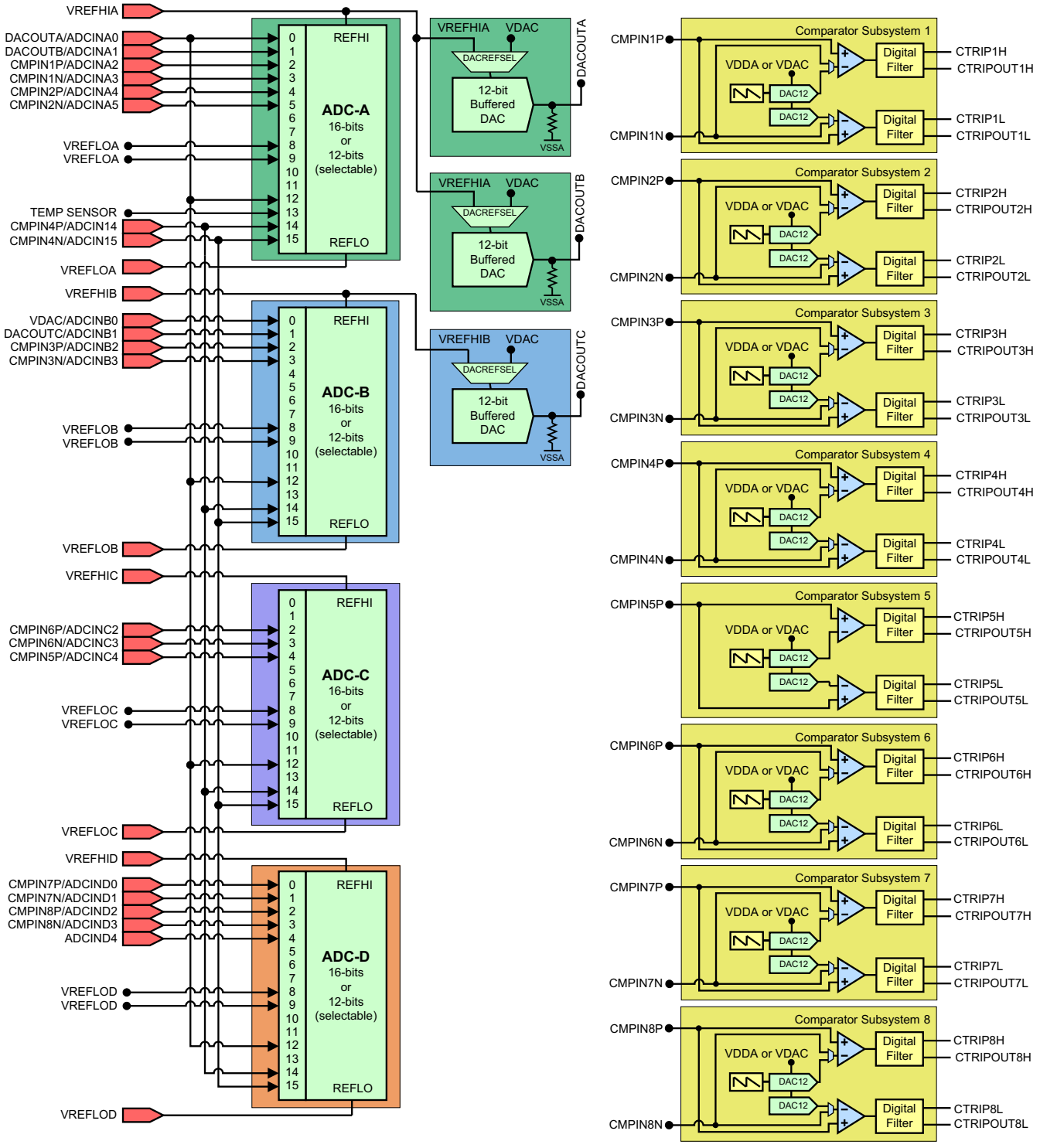



**Figure 10-2. Analog Subsystem Block Diagram (176-Pin PTP)**



Copyright © 2024 Texas Instruments Incorporated

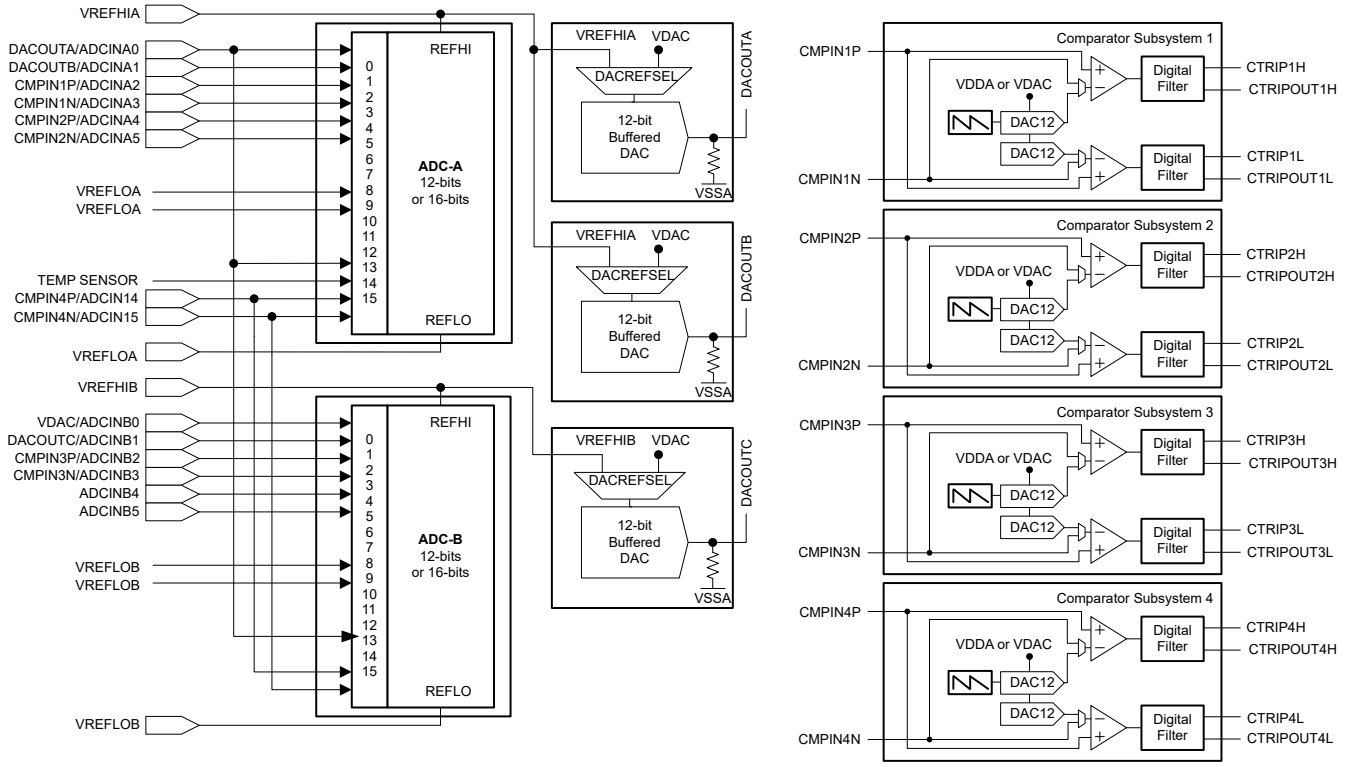



**Figure 10-3. Analog Subsystem Block Diagram (100-Pin PZP)**


Copyright © 2024 Texas Instruments Incorporated

**Table 10-1. Analog Signal Descriptions**

| Signal Name | Description |
|---|---|
| ADCINAx | ADC A Input |
| ADCINBx | ADC B Input |
| ADCINCx | ADC C Input |
| ADCINDx | ADC D Input |
| CMPINxP | Comparator subsystem positive input |
| CMPINxN | Comparator subsystem negative input |
| DACOUTx | Buffered DAC Output |
| TEMP SENSOR | Internal temperature sensor |
| VDAC | Optional external reference voltage for on-chip DACs. There is a 100pF capacitor to VSSA on this pin whether used for ADC input or DAC reference that cannot be disabled. If this pin is being used as a reference for the on-chip DACs, place at least a 1µF capacitor on this pin. |

**Table 10-2. Reference Summary**

| Module | Reference Option | Configured Where? | Register | Driverlib Function | Notes |
|---|---|---|---|---|---|
| ADC | External or Internal | Not Configurable | N/A | N/A | VREFHI must always be driven externally on this device. |
| Buffered DAC | VREFHI or VDAC | DAC Module | DacxRegs. DACCTL.bit.DACREFSEL | DAC_CTL_DACREFSEL | |
| | External or Internal | Not Configurable | N/A | N/A | VREFHI must always be driven externally on this device. |
| CMPSS DACs | VDDA or VDAC | CMPSS Module | CmpssxRegs. COMPDACCTL.bit.SELREF | CMPSS_COMPDACCTL_SELREF | |

## 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 $t_{ADCPUINT}$ to power up in internal reference mode, and $t_{ADCPUEXT}$ to power up in external reference mode, the application does not need to wait $t_{ADCPUINT}$ 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 $t_{DACPUINT}$.

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.



Copyright © 2024 Texas Instruments Incorporated

## 10.3 Analog Subsystem Registers

This section describes the Analog Subsystem Registers.

### *10.3.1 Analog Subsystem Base Addresses*

**Table 10-3. Analog Subsystem Base Address Table**

| Device Registers | Register Name | Start Address | End Address |
|---|---|---|---|
| AnalogSubsysRegs | ANALOG_SUBSYS_REGS | 0x0005_D180 | 0x0005_D1FF |



Copyright © 2024 Texas Instruments Incorporated

### 10.3.2 ANALOG_SUBSYS_REGS Registers

Table 10-4 lists the memory-mapped registers for the ANALOG_SUBSYS_REGS registers. All register offset addresses not listed in Table 10-4 should be considered as reserved locations and the register contents should not be modified.

**Table 10-4. ANALOG_SUBSYS_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 20h | INTOSC1TRIM | Internal Oscillator 1 Trim Register | EALLOW | Go |
| 22h | INTOSC2TRIM | Internal Oscillator 2 Trim Register | EALLOW | Go |
| 26h | TSNSCTL | Temperature Sensor Control Register | EALLOW | Go |
| 2Eh | LOCK | Lock Register | EALLOW | Go |
| 36h | ANAREFTRIMA | Analog Reference Trim A Register | EALLOW | Go |
| 38h | ANAREFTRIMB | Analog Reference Trim B Register | EALLOW | Go |
| 3Ah | ANAREFTRIMC | Analog Reference Trim C Register | EALLOW | Go |
| 3Ch | ANAREFTRIMD | Analog Reference Trim D Register | EALLOW | Go |

Complex bit access types are encoded to fit into small table cells. Table 10-5 shows the codes that are used for access types in this section.

**Table 10-5. ANALOG_SUBSYS_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| R-0 | R<br>-0 | Read<br>Returns 0s |
| Write Type | | |
| W | W | Write |
| WOnce | W<br>Once | Write<br>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. |



Copyright © 2024 Texas Instruments Incorporated

### 10.3.2.1 INTOSC1TRIM Register (Offset = 20h) [Reset = 00000000h]

INTOSC1TRIM is shown in Figure 10-4 and described in Table 10-6.

Return to the Summary Table.

Internal Oscillator 1 Trim Register

**Figure 10-4. INTOSC1TRIM Register**

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

**Table 10-6. INTOSC1TRIM Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-24 | RESERVED | R | 0h | Reserved |
| 23-16 | RESERVED | R/W | 0h | Reserved |
| 15-12 | RESERVED | R | 0h | Reserved |
| 11-0 | VALFINETRIM | R/W | 0h | Oscillator Value Fine Trim Bits.<br>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.<br>Reset type: XRSn |



Copyright © 2024 Texas Instruments Incorporated

### 10.3.2.2 INTOSC2TRIM Register (Offset = 22h) [Reset = 00000000h]

INTOSC2TRIM is shown in Figure 10-5 and described in Table 10-7.

Return to the Summary Table.

Internal Oscillator 2 Trim Register

**Figure 10-5. INTOSC2TRIM Register**

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

**Table 10-7. INTOSC2TRIM Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-24 | RESERVED | R | 0h | Reserved |
| 23-16 | RESERVED | R/W | 0h | Reserved |
| 15-12 | RESERVED | R | 0h | Reserved |
| 11-0 | VALFINETRIM | R/W | 0h | Oscillator Value Fine Trim Bits.<br>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.<br>Reset type: XRSn |



Copyright © 2024 Texas Instruments Incorporated

### 10.3.2.3 TSNSCTL Register (Offset = 26h) [Reset = 0000h]

TSNSCTL is shown in Figure 10-6 and described in Table 10-8.

Return to the Summary Table.

Temperature Sensor Control Register

**Figure 10-6. TSNSCTL Register**

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

**Table 10-8. TSNSCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-1 | RESERVED | R | 0h | Reserved |
| 0 | ENABLE | R/W | 0h | Temperature Sensor Enable. This bit enables the temperature sensor output to the ADC.<br>0 Disabled<br>1 Enabled<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 10.3.2.4 LOCK Register (Offset = 2Eh) [Reset = 00000000h]

LOCK is shown in Figure 10-7 and described in Table 10-9.

Return to the Summary Table.

Lock Register

**Figure 10-7. LOCK Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | ANAREFTRIMD | ANAREFTRIMC | ANAREFTRIMB |
| R-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 |
| ANAREFTRIMA | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | | |
| R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R-0h | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | RESERVED | RESERVED | RESERVED | TSNSCTL | RESERVED | RESERVED | RESERVED |
| R-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h |

**Table 10-9. LOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | RESERVED | R | 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 | ANAREFTRIMD | R/WOnce | 0h | Analog Reference D Trim Register Lock. Setting this bit will disable any future write to the respective register. This bit can only be cleared by a reset.<br>Reset type: CPU1.SYSRSn |
| 25 | ANAREFTRIMC | R/WOnce | 0h | Analog Reference C Trim Register Lock. Setting this bit will disable any future write to the respective register. This bit can only be cleared by a reset.<br>Reset type: CPU1.SYSRSn |
| 24 | ANAREFTRIMB | R/WOnce | 0h | Analog Reference B Trim Register Lock. Setting this bit will disable any future write to the respective register. This bit can only be cleared by a reset.<br>Reset type: CPU1.SYSRSn |
| 23 | ANAREFTRIMA | R/WOnce | 0h | Analog Reference A Trim Register Lock. Setting this bit will disable any future write to the respective register. This bit can only be cleared by a reset.<br>Reset type: CPU1.SYSRSn |
| 22 | RESERVED | R/WOnce | 0h | Reserved |
| 21 | RESERVED | R/WOnce | 0h | Reserved |
| 20 | RESERVED | R/WOnce | 0h | Reserved |
| 19 | RESERVED | R/WOnce | 0h | Reserved |
| 18-7 | RESERVED | R | 0h | Reserved |
| 6 | RESERVED | R/WOnce | 0h | Reserved |
| 5 | RESERVED | R/WOnce | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 10-9. LOCK Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 4 | RESERVED | R/WOnce | 0h | Reserved |
| 3 | TSNSCTL | R/WOnce | 0h | Temperature Sensor Control Register Lock. Setting this bit will disable any future write to the respective register. This bit can only be cleared by a reset.<br>Reset type: CPU1.SYSRSn |
| 2 | RESERVED | R/WOnce | 0h | Reserved |
| 1 | RESERVED | R/WOnce | 0h | Reserved |
| 0 | RESERVED | R/WOnce | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 10.3.2.5 ANAREFTRIMA Register (Offset = 36h) [Reset = 00000000h]

ANAREFTRIMA is shown in Figure 10-8 and described in Table 10-10.

Return to the Summary Table.

Analog Reference Trim A Register

**Figure 10-8. ANAREFTRIMA Register**

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

**Table 10-10. ANAREFTRIMA Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-24 | RESERVED | R | 0h | Reserved |
| 23-16 | RESERVED | R/W | 0h | Reserved |
| 15-11 | IREFTRIM | R/W | 0h | Reference Current Trim. This bit field defines the reference current trim value.<br>0x0 - Untrimmed<br>all other values - Trimmed<br>Reset type: XRSn |
| 10-6 | BGSLOPETRIM | R/W | 0h | Bandgap Slope Trim. This bit field defines the bandgap slope trim value.<br>0x0 - Untrimmed<br>all other values - Trimmed<br>Reset type: XRSn |
| 5-0 | BGVALTRIM | R/W | 0h | Bandgap Value Trim. This bit field defines the bandgap voltage offset trim value.<br>0x0 - Untrimmed<br>all other values - Trimmed<br>Reset type: XRSn |



Copyright © 2024 Texas Instruments Incorporated

### 10.3.2.6 ANAREFTRIMB Register (Offset = 38h) [Reset = 00000000h]

ANAREFTRIMB is shown in Figure 10-9 and described in Table 10-11.

Return to the Summary Table.

Analog Reference Trim B Register

**Figure 10-9. ANAREFTRIMB Register**

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

**Table 10-11. ANAREFTRIMB Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-24 | RESERVED | R | 0h | Reserved |
| 23-16 | RESERVED | R/W | 0h | Reserved |
| 15-11 | IREFTRIM | R/W | 0h | Reference Current Trim. This bit field defines the reference current trim value.<br>0x0 - Untrimmed<br>all other values - Trimmed<br>Reset type: XRSn |
| 10-6 | BGSLOPETRIM | R/W | 0h | Bandgap Slope Trim. This bit field defines the bandgap slope trim value.<br>0x0 - Untrimmed<br>all other values - Trimmed<br>Reset type: XRSn |
| 5-0 | BGVALTRIM | R/W | 0h | Bandgap Value Trim. This bit field defines the bandgap voltage offset trim value.<br>0x0 - Untrimmed<br>all other values - Trimmed<br>Reset type: XRSn |



Copyright © 2024 Texas Instruments Incorporated

### 10.3.2.7 ANAREFTRIMC Register (Offset = 3Ah) [Reset = 00000000h]

ANAREFTRIMC is shown in Figure 10-10 and described in Table 10-12.

Return to the Summary Table.

Analog Reference Trim C Register

**Figure 10-10. ANAREFTRIMC Register**

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

**Table 10-12. ANAREFTRIMC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-24 | RESERVED | R | 0h | Reserved |
| 23-16 | RESERVED | R/W | 0h | Reserved |
| 15-11 | IREFTRIM | R/W | 0h | Reference Current Trim. This bit field defines the reference current trim value.<br>0x0 - Untrimmed<br>all other values - Trimmed<br>Reset type: XRSn |
| 10-6 | BGSLOPETRIM | R/W | 0h | Bandgap Slope Trim. This bit field defines the bandgap slope trim value.<br>0x0 - Untrimmed<br>all other values - Trimmed<br>Reset type: XRSn |
| 5-0 | BGVALTRIM | R/W | 0h | Bandgap Value Trim. This bit field defines the bandgap voltage offset trim value.<br>0x0 - Untrimmed<br>all other values - Trimmed<br>Reset type: XRSn |



Copyright © 2024 Texas Instruments Incorporated

### 10.3.2.8 ANAREFTRIMD Register (Offset = 3Ch) [Reset = 00000000h]

ANAREFTRIMD is shown in Figure 10-11 and described in Table 10-13.

Return to the Summary Table.

Analog Reference Trim D Register

**Figure 10-11. ANAREFTRIMD Register**

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

**Table 10-13. ANAREFTRIMD Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-24 | RESERVED | R | 0h | Reserved |
| 23-16 | RESERVED | R/W | 0h | Reserved |
| 15-11 | IREFTRIM | R/W | 0h | Reference Current Trim. This bit field defines the reference current trim value.<br>0x0 - Untrimmed<br>all other values - Trimmed<br>Reset type: XRSn |
| 10-6 | BGSLOPETRIM | R/W | 0h | Bandgap Slope Trim. This bit field defines the bandgap slope trim value.<br>0x0 - Untrimmed<br>all other values - Trimmed<br>Reset type: XRSn |
| 5-0 | BGVALTRIM | R/W | 0h | Bandgap Value Trim. This bit field defines the bandgap voltage offset trim value.<br>0x0 - Untrimmed<br>all other values - Trimmed<br>Reset type: XRSn |



Copyright © 2024 Texas Instruments Incorporated

Chapter 11

# Analog-to-Digital Converter (ADC)

The analog-to-digital converter (ADC) module described in this chapter is a Type 4 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** ..... 1583
**11.2 ADC Configurability** ..... 1586
**11.3 SOC Principle of Operation** ..... 1590
**11.4 SOC Configuration Examples** ..... 1594
**11.5 ADC Conversion Priority** ..... 1596
**11.6 Burst Mode** ..... 1599
**11.7 EOC and Interrupt Operation** ..... 1601
**11.8 Post-Processing Blocks** ..... 1603
**11.9 Opens/Shorts Detection Circuit (OSDETECT)** ..... 1607
**11.10 Power-Up Sequence** ..... 1609
**11.11 ADC Calibration** ..... 1609
**11.12 ADC Timings** ..... 1612
**11.13 Additional Information** ..... 1618
**11.14 Software** ..... 1627
**11.15 ADC Registers** ..... 1630





Copyright © 2024 Texas Instruments Incorporated

## 11.1 Introduction

The ADC module is a successive approximation (SAR) style ADC with selectable resolution of either 16 bits or 12 bits. 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**

- *ADC Oversampling Application Report*
- 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



Copyright © 2024 Texas Instruments Incorporated

- 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:

- Selectable resolution of 12 bits or 16 bits
- Ratiometric external reference set by VREFHI and VREFLO pins
- Differential signal conversions (16-bit mode only)
- Single-ended signal conversions (12-bit mode only)
- Input multiplexer with up to 16 channels (single-ended) or 8 channels (differential)
- 16 configurable 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 (from each C28x core present)
  - 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.



Copyright © 2024 Texas Instruments Incorporated

### 11.1.3 Block Diagram

Figure 11-1 shows the block diagram for the ADC core and ADC wrapper.

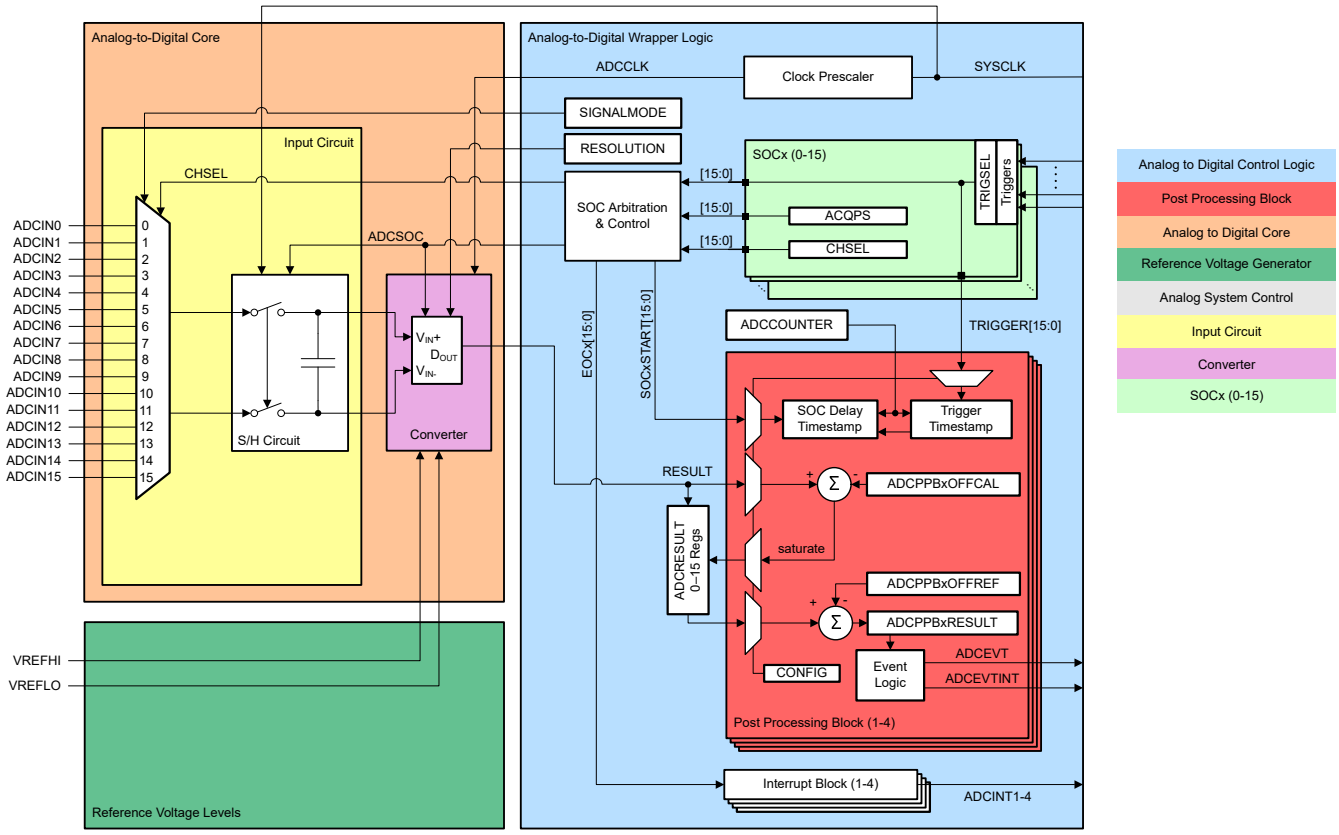



**Figure 11-1. ADC Module Block Diagram**

**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.



Copyright © 2024 Texas Instruments Incorporated

## 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(1) |
| Resolution | Per module(1) |
| Signal mode | Per module |
| Reference voltage source | Not configurable (external reference only) |
| Trigger source | Per SOC(1) |
| Converted channel | Per SOC |
| Acquisition window duration | Per SOC(1) |
| EOC location | Per module (early or late) |
| Burst Mode | Per module(1) |

(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.

In 16-bit mode, the core requires approximately 29.5 ADCCLK cycles to process a voltage into a conversion result, while in 12-bit mode, this process requires approximately 10.5 ADCCLK cycles. The choice of resolution also determines the necessary 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 configurable resolution of 16 bits or 12 bits.

The resolution must be configured by using either the AdcSetMode() or ADC_setMode() functions, depending on the header files used, provided in C2000ware in F2837xD_Adc.c. These functions make sure that the correct trim is loaded into the ADC trim registers, and must be called at least once after a device reset. Do not configure the resolution by directly writing to the ADCCTL2 register.

The resolution can be changed at any time when the ADC is idle (no active or pending SOCs). No wait time is necessary after changing the resolution before conversions can be initiated. If SOCs are active or pending when the resolution is changed, those SOCs can produce incorrect conversion results.



Copyright © 2024 Texas Instruments Incorporated

### 11.2.3 Voltage Reference

#### 11.2.3.1 External Reference Mode

Each ADC has a VREFHI input and a VREFLO input. In external reference mode, these pins are used as a ratiometric reference to determine the ADC conversion input range.

See Section 11.13.6 for information on how to supply the reference voltage.

**Note**

- On devices with no external VREFLO pin, VREFLO is internally connected to the device analog ground, VSSA.
- 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.

### 11.2.4 Signal Mode

The ADC supports two signal modes: single-ended and differential.

In single-ended mode, the input voltage to the converter is sampled through a single pin (ADCINx), referenced to VREFLO.

In differential signaling mode, the input voltage to the converter is sampled through a pair of input pins, one of which is the positive input (ADCINxP) and the other is the negative input (ADCINxN). The actual input voltage is the difference between the two (ADCINxP – ADCINxN).

**Note**

- In differential signal mode, VREFLO must be connected to VSSA.
- In differential signal mode, the common mode voltage is $V_{CM}$ = (ADCINxP + ADCINxN)/2

  The data sheet for a particular device places some requirements on how close this voltage needs to be to: (VREFHI + VREFLO)/2

  **Note**: The above condition is not met by connecting the negative input to VSSA or VREFLO.
- Differential signaling mode is advantageous because noise encountered on both inputs is largely canceled. The effect can be maximized by routing the positive and negative traces for the same differential input as close together as possible and keeping them symmetrical with respect to the signal reference.

The signal mode must be configured by using either the ADC_setMode() or AdcSetMode() functions, depending on the header files used, provided in C2000warein F2837xD_Adc.c. These functions make sure that the correct trim is loaded into the ADC trim registers. These functions must be called at least once after a device reset. The signal mode must not be configured by writing to the ADCCTL2 register directly.



Copyright © 2024 Texas Instruments Incorporated

### 11.2.5 Expected Conversion Results

Based on a given analog input voltage, the expected digital conversion is given in Table 11-2 and Table 11-3. Fractional values are truncated.

**Table 11-2. Analog to 12-bit Digital Formulas**

| | Analog Input | Digital Result |
|---|---|---|
| **Single-Ended** | when ADCINy ≤ VREFLO | ADCRESULTx = 0 |
| | when VREFLO < ADCINy < VREFHI | $ADCRESULTx = 4096\left(\frac{ADCINy - VREFLO}{VREFHI - VREFLO}\right)$ |
| | when ADCINy ≥ VREFHI | ADCRESULTx = 4095 |
| **Differential** | when ADCINyP - ADCINyN ≤ -(VREFHI-VREFLO) | ADCRESULTx = 0 |
| | when -(VREFHI-VREFLO) < ADCINyP - ADCINyN ≤ (VREFHI-VREFLO) | $ADCRESULTx = 4096\left(\frac{ADCINyP - ADCINyN + VREFHI - VREFLO}{2(VREFHI - VREFLO)}\right)$ |
| | when ADCINyP - ADCINyN ≥ (VREFHI-VREFLO) | ADCRESULTx = 4095 |

**Table 11-3. Analog to 16-bit Digital Formulas**

| | Analog Input | Digital Result |
|---|---|---|
| **Single-Ended** | Invalid Mode | Invalid Mode |
| **Differential** | when ADCINyP - ADCINyN ≤ -(VREFHI-VREFLO) | ADCRESULTx = 0 |
| | when -(VREFHI-VREFLO) < ADCINyP - ADCINyN ≤ (VREFHI-VREFLO) | $ADCRESULTx = 65536\left(\frac{ADCINy - VREFLO}{VREFHI - VREFLO}\right)$ |
| | when ADCINyP - ADCINyN ≥ (VREFHI-VREFLO) | ADCRESULTx = 65535 |



Copyright © 2024 Texas Instruments Incorporated

### 11.2.6 Interpreting Conversion Results

Based on a given ADC conversion result, the corresponding analog input is given in Table 11-4 and Table 11-5. This corresponds to the center of the possible range of analog voltages that can produce this conversion result.

**Table 11-4. 12-Bit Digital-to-Analog Formulas**

| | Digital Value | Analog Equivalent |
|---|---|---|
| **Single-Ended** | when ADCRESULTy = 0 | $\text{ADCINx} \leq \text{VREFLO}$ (1) |
| | when 0 < ADCRESULTy < 4095 | $\text{ADCINx} = (\text{VREFHI} - \text{VREFLO})\left(\frac{\text{ADCRESULTy}}{4096}\right) + \text{VREFLO}$ (2) |
| | when ADCRESULTy = 4095 | $\text{ADCINx} \geq \text{VREFHI}$ (3) |
| **Differential** | when ADCRESULTy = 0 | $ADCINxP - ADCINxN \leq (VREFHI - VREFLO)$ (4) |
| | when 0 < ADCRESULTy < 4095 | $ADCINxP - ADCINxN = (VREFHI - VREFLO)\left(\frac{ADCRESULTy}{2048} - 1\right)$ (5) |
| | when ADCRESULTy = 4095 | $ADCINxP - ADCINxN \geq (VREFHI - VREFLO)$ (6) |

**Table 11-5. 16-Bit Digital-to-Analog Formulas**

| | Digital Value | Analog Equivalent |
|---|---|---|
| **Single-Ended** | Invalid Mode | Invalid Mode |
| **Differential** | when ADCRESULTy = 0 | $ADCINxP - ADCINxN \leq (VREFHI - VREFLO)$ (7) |
| | when 0 < ADCRESULTy < 65535 | $ADCINxP - ADCINxN = (VREFHI - VREFLO)\left(\frac{ADCRESULTy}{2048} - 1\right)$ (8) |
| | when ADCRESULTy = 65535 | $ADCINxP - ADCINxN \geq (VREFHI - VREFLO)$ (9) |



Copyright © 2024 Texas Instruments Incorporated

## 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.

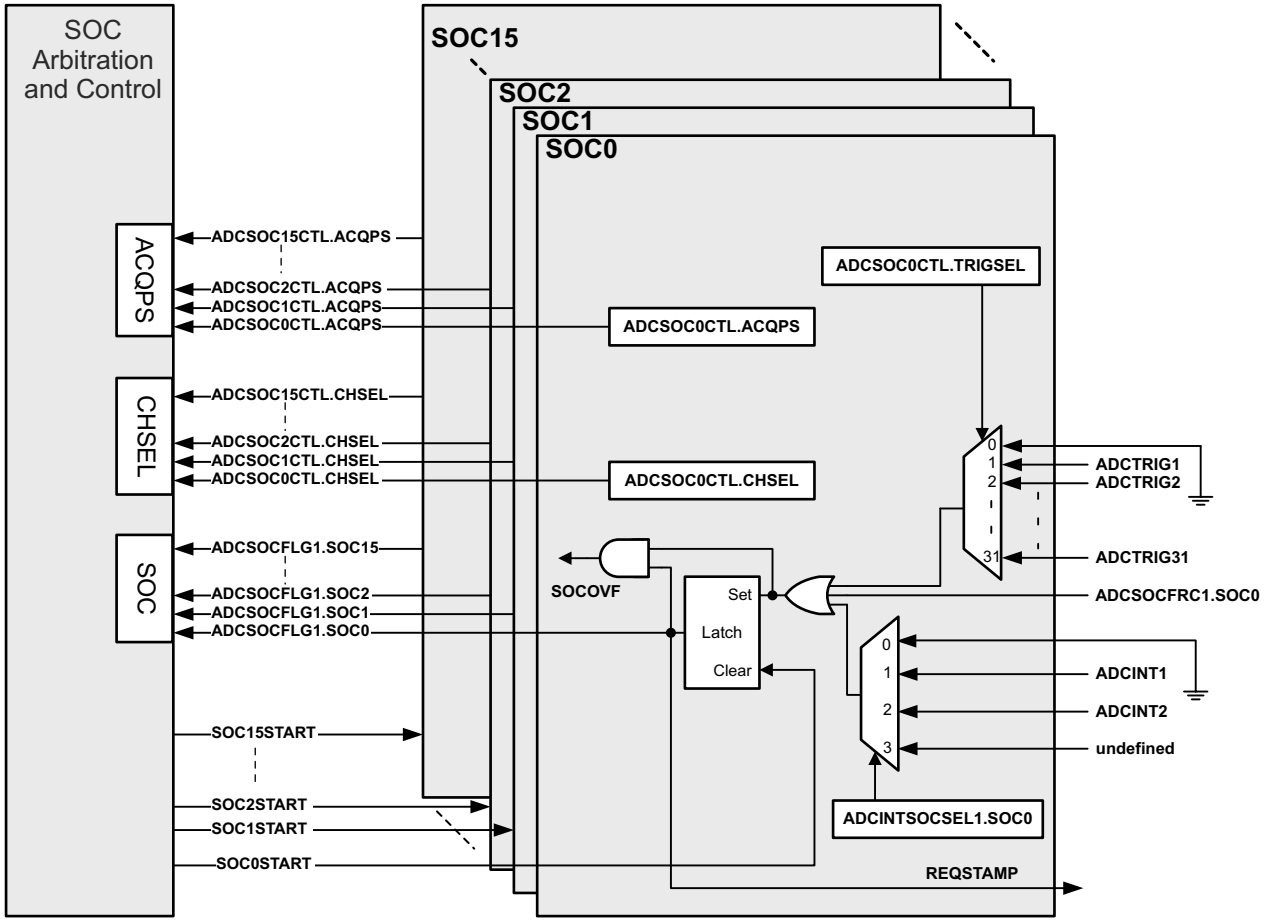



**Figure 11-2. SOC Block Diagram**



Copyright © 2024 Texas Instruments Incorporated

### 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 (from each C28x core present)
- 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.

### 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:

Acquisition window = (ACQPS + 1) * (System Clock (SYSCLK) cycle time)

- The acquisition window duration is based on the System Clock (SYSCLK), not the ADC clock (ADCCLK).
- The selected acquisition window duration must be at least as long as one ADCCLK cycle.
- The data sheet specifies a minimum acquisition window duration (in nanoseconds). The user is responsible for selecting an acquisition window duration that meets this requirement.



Copyright © 2024 Texas Instruments Incorporated

### 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.

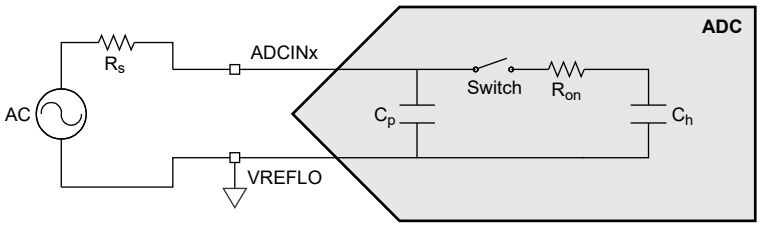



**Figure 11-3. Single-Ended Input Model**

For differential operation, the ADC input characteristics for values in the differential input model (see Figure 11-4 ) can be found in the device data sheet.

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.

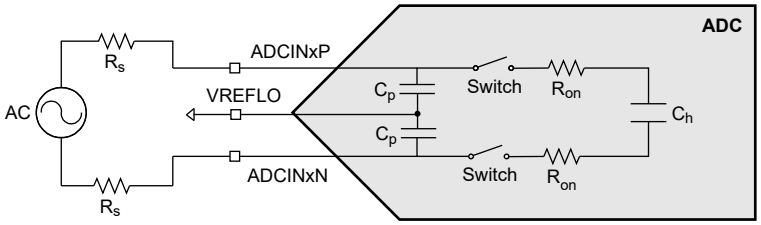



**Figure 11-4. Differential Input Model**



Copyright © 2024 Texas Instruments Incorporated

### 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. Depending on the signal mode, the selection is different. For single-ended signal mode, the value in CHSEL selects a single pin as the input. For differential signal mode, the value in CHSEL selects an even-odd pin pair to be the positive and negative inputs. This is summarized in Table 11-6.

**Note**

Regardless of configured resolution and signal mode, channel 13 on ADC-A (temperature sensor) and channel 12 on all ADCs are sampled in 12-bit single-ended mode.

**Table 11-6. Channel Selection of Input Pins**

| Input Mode | CHSEL | Input | |
|---|---|---|---|
| Single-Ended | 0 | ADCIN0 | |
| | 1 | ADCIN1 | |
| | 2 | ADCIN2 | |
| | 3 | ADCIN3 | |
| | 4 | ADCIN4 | |
| | 5 | ADCIN5 | |
| | 6 | ADCIN6 | |
| | 7 | ADCIN7 | |
| | 8 | ADCIN8 | |
| | 9 | ADCIN9 | |
| | 10 | ADCIN10 | |
| | 11 | ADCIN11 | |
| | 12 | ADCIN12 | |
| | 13 | ADCIN13 | |
| | 14 | ADCIN14 | |
| | 15 | ADCIN15 | |
| | **CHSEL** | **Positive Input** | **Negative Input** |
| Differential | 0 or 1 | ADCIN0 | ADCIN1 |
| | 2 or 3 | ADCIN2 | ADCIN3 |
| | 4 or 5 | ADCIN4 | ADCIN5 |
| | 6 or 7 | ADCIN6 | ADCIN7 |
| | 8 or 9 | ADCIN8 | ADCIN9 |
| | 10 or 11 | ADCIN10 | ADCIN11 |
| | 12 or 13 | ADCIN12 | ADCIN13 |
| | 14 or 15 | ADCIN14 | ADCIN15 |



Copyright © 2024 Texas Instruments Incorporated

## 11.4 SOC Configuration Examples

The following sections provide some specific examples of how to configure the SOCs to produce some conversions.

### *11.4.1 Single Conversion from ePWM Trigger*

To configure ADCA to perform a single conversion on channel ADCIN1 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 SOCs can be used.

Assuming a 100ns sample window is desired with a SYSCLK frequency of 200MHz, then the acquisition window duration must be 100ns/5ns = 20 SYSCLK cycles. The ACQPS field must be set to 20 - 1 = 19.

```
AdcaRegs.ADCSOC5CTL.bit.CHSEL = 1;      //SOC5 converts ADCINA1
AdcaRegs.ADCSOC5CTL.bit.ACQPS = 19;     //SOC5 uses a sample duration of 20 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 = 19;       //SOC5 uses a sample duration of 20 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 = 19;       //SOC6 uses a sample duration of 20 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 = 19;       //SOC7 uses a sample duration of 20 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 = 19;       //SOC8 uses a sample duration of 20 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.



Copyright © 2024 Texas Instruments Incorporated

### 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 11-7, where a SYCLK of 200MHz is assumed (5ns cycle time).

**Table 11-7. Example Requirements for Multiple Signal Sampling**

| Signal Name | Acquisition Window Requirement | Acquisition Window (SYSCLK Cycles) | ACQPS Register Value |
|---|---|---|---|
| Signal 1 | >120ns | 120ns/5ns = 24 | 24 – 1 = 23 |
| Signal 2 | >444ns | 444ns/5ns = 89 (round up) | 89 – 1 = 88 |
| Signal 3 | >110ns | 110ns/5ns = 22 | 22 – 1 = 21 |
| Signal 4 | >291ns | 291ns/5ns = 59 (round up) | 59 – 1 = 58 |

Next decide which ADC pins to connect to each signal. This is highly dependent on the application board layout. Once the pins are selected, determining the value of CHSEL is straightforward (see Table 11-8).

**Table 11-8. Example Connections for Multiple Signal Sampling**

| Signal Name | ADC Pin | CHSEL Register Value |
|---|---|---|
| Signal 1 | ADCINA5 | 5 |
| Signal 2 | ADCINA0 | 0 |
| Signal 3 | ADCINA3 | 3 |
| Signal 4 | ADCINA2 | 2 |

With the information tabulated, 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.



Copyright © 2024 Texas Instruments Incorporated

### 11.4.4 Software Triggering of SOCs

At any point, whether or not the SOCs have been configured to accept a specific trigger, a software trigger can set the SOCs 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.all = 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 SOCPRICTL 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 11-5.

The SOCPRIORITY field in the ADCSOCPRIORITYCTL register can be used to assign high priority from a single to all of the SOCs. 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 SOCs 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 11-6.



Copyright © 2024 Texas Instruments Incorporated

**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 .

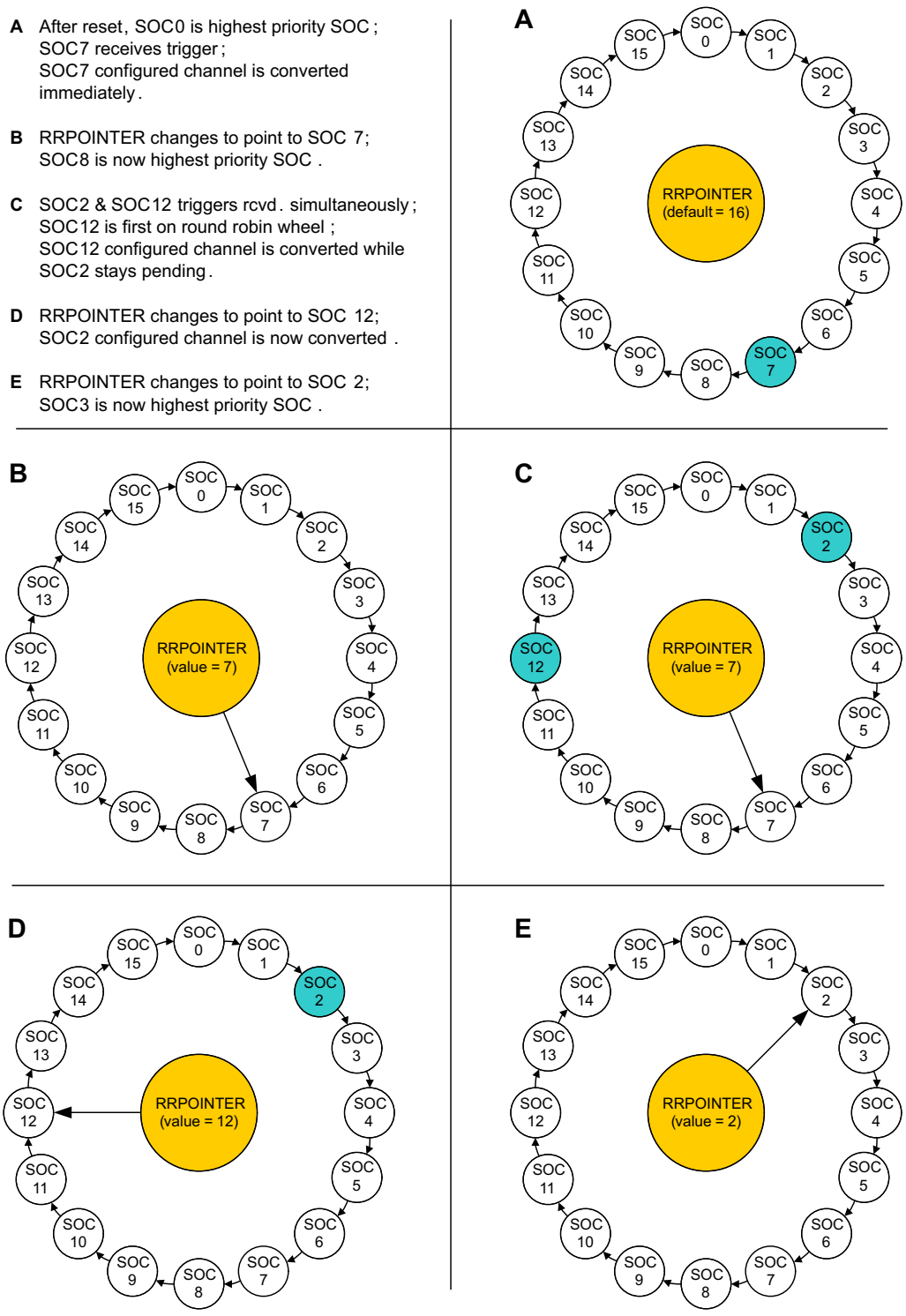



**Figure 11-5. Round Robin Priority Example**


Copyright © 2024 Texas Instruments Incorporated

Example when SOCPRIORITY = 4

**A** After reset, SOC4 is 1st on round robin wheel ;
SOC7 receives trigger ;
SOC7 configured channel is converted immediately .

**B** RRPOINTER changes to point to SOC 7;
SOC8 is now 1st on round robin wheel .

**C** SOC2 & SOC12 triggers rcvd . simultaneously ;
SOC2 interrupts round robin wheel and SOC 2 configured channel is converted while SOC 12 stays pending .

**D** RRPOINTER stays pointing to 7;
SOC12 configured channel is now converted .

**E** RRPOINTER changes to point to SOC 12;
SOC13 is now 1st on round robin wheel .

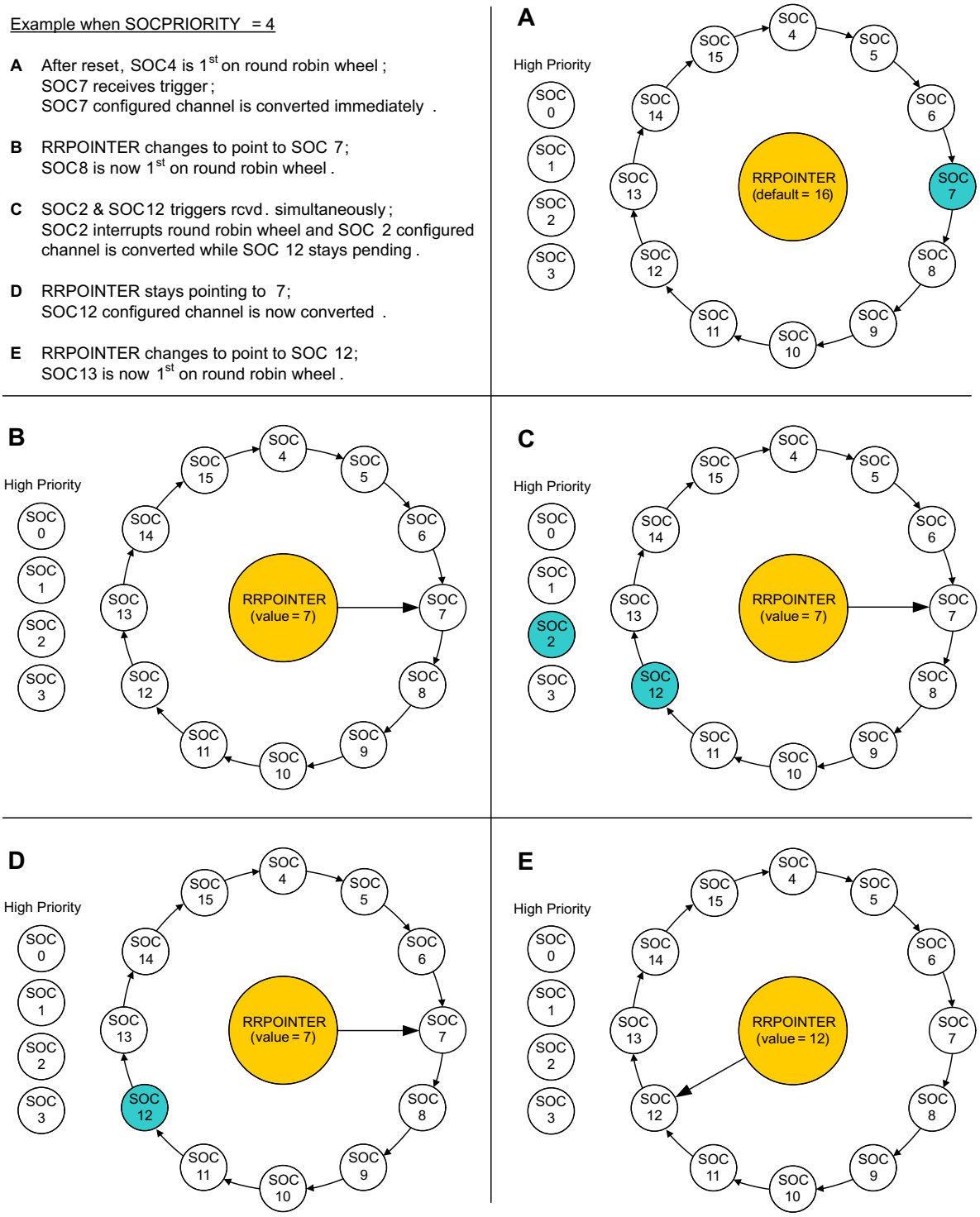



**Figure 11-6. High Priority Example**



Copyright © 2024 Texas Instruments Incorporated

## 11.6 Burst Mode

Burst mode allows a single trigger to walk through the round-robin SOCs 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 SOCs that are configured for round-robin operation (not high priority). Instead of the TRIGSEL field, all round-robin SOCs 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 SOCs to be converted, but only (ADCBURSTCTL.BURSTSIZE + 1) SOCs. The first SOC to be set is the SOC with the highest priority based on the round-robin pointer, and subsequent SOCs are set until BURSTSIZE SOCs 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 SOCs 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 ≤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.SOCPRICTL.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 SOCs 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.



Copyright © 2024 Texas Instruments Incorporated

### 11.6.2 Burst Mode Priority Example

An example of priority resolution using burst mode and high-priority SOCs is presented in Figure 11-7.

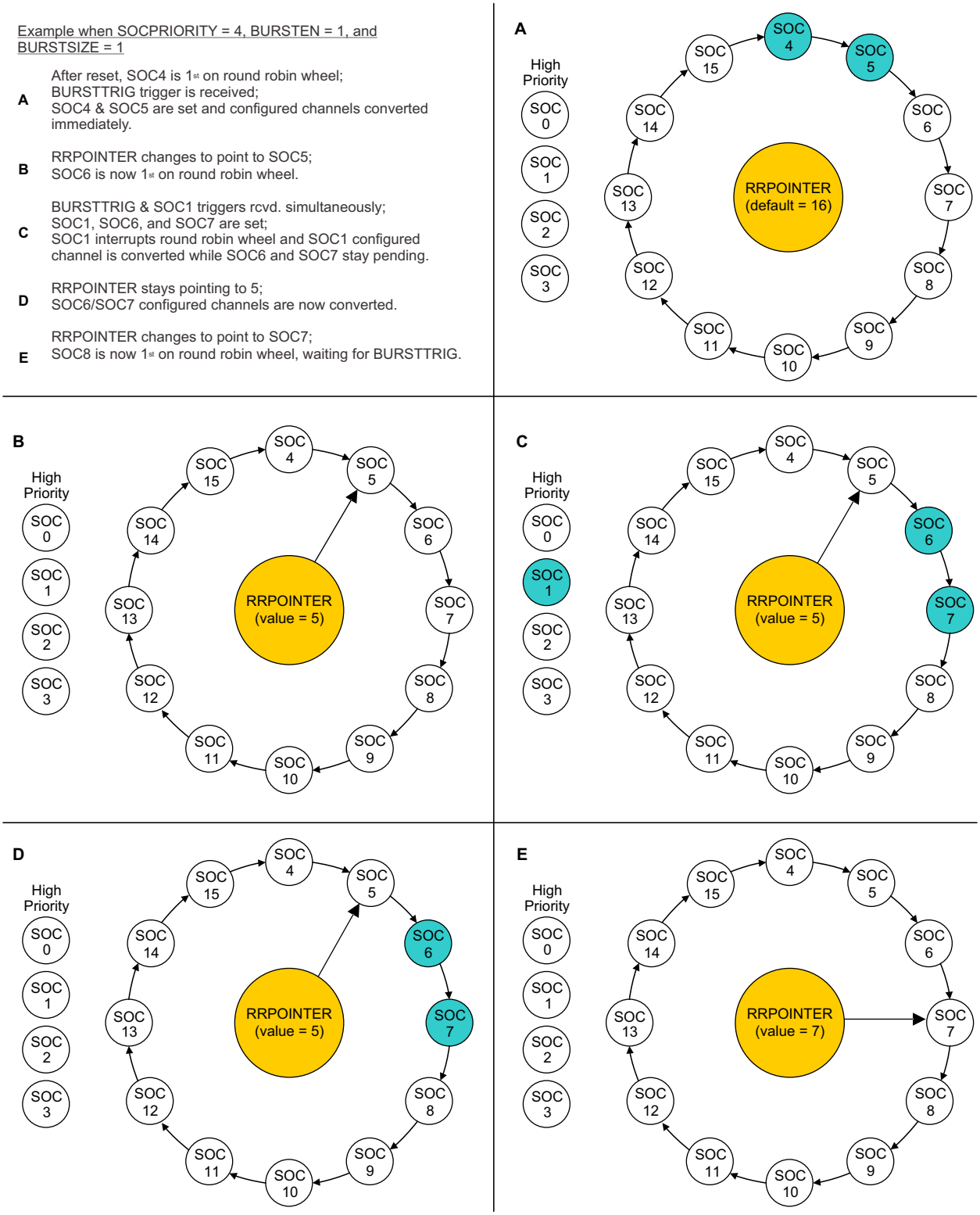



**Figure 11-7. Burst Priority Example**



Copyright © 2024 Texas Instruments Incorporated

## 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 11-8 shows a block diagram of the ADC interrupt structure.

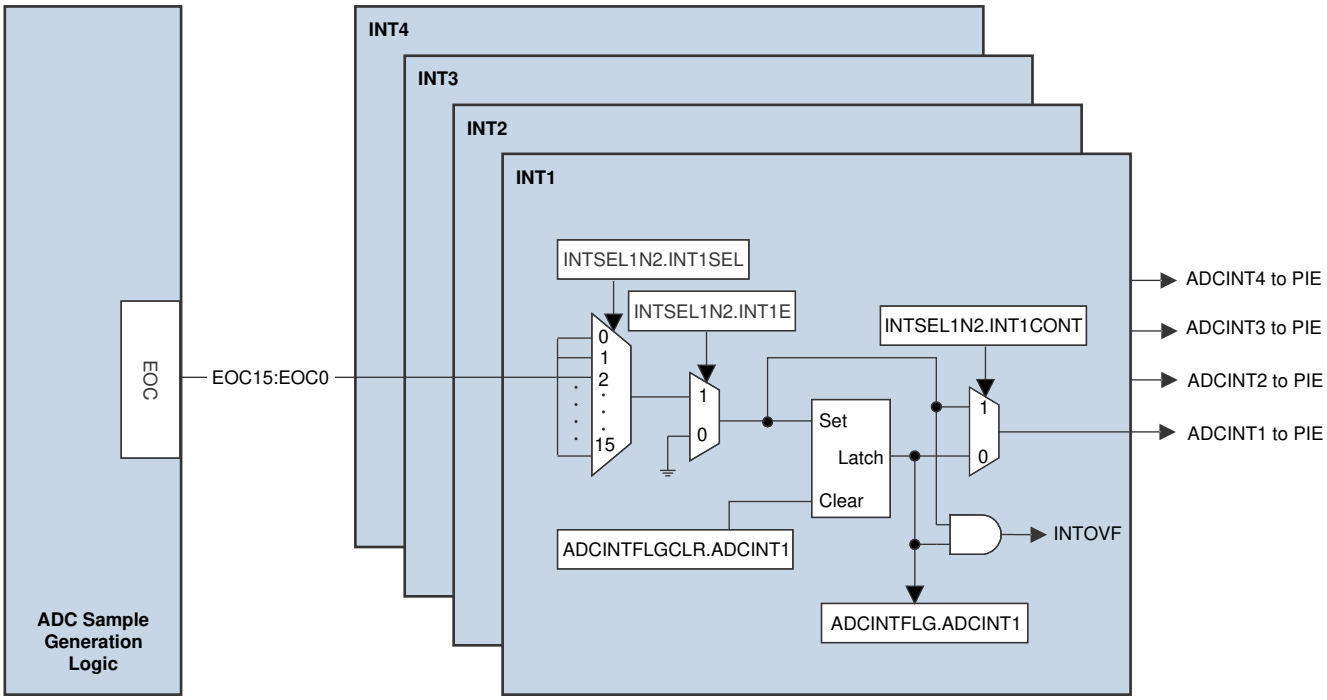



**Figure 11-8. ADC EOC Interrupts**



Copyright © 2024 Texas Instruments Incorporated

### 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 ADCINOVF 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.



Copyright © 2024 Texas Instruments Incorporated

## 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 11-9 presents the structure of each PPB. Subsequent sections explain the use of each submodule.

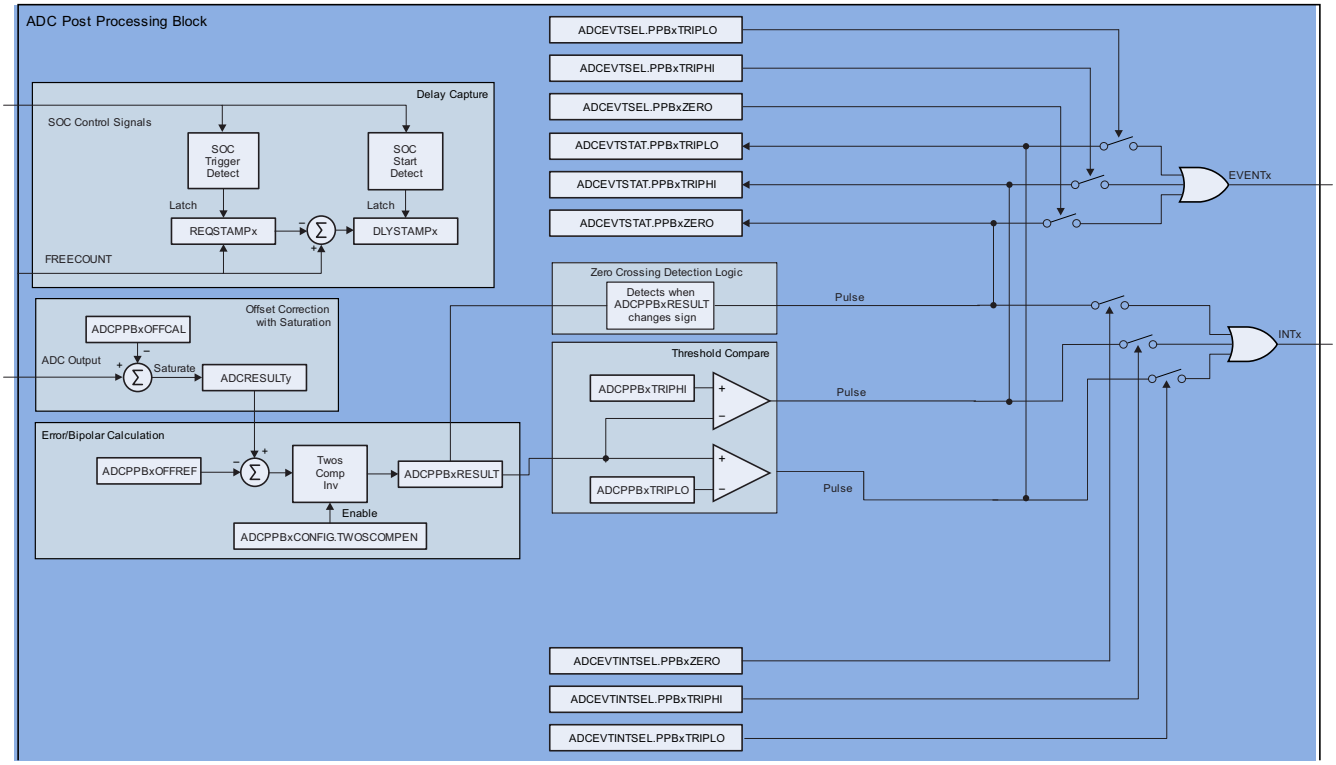



**Figure 11-9. ADC PPB Block Diagram**



Copyright © 2024 Texas Instruments Incorporated

### 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 either 4095 or 65535 on the high end for 12-bit or 16-bit mode, respectively.

**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**

- In 12-bit mode, 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



Copyright © 2024 Texas Instruments Incorporated

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 11-10.

**Note**

- Zero-crossing and limit compare reference the ADCPPBxRESULT register. This includes any correction applied by the OFFCAL and OFFREF registers. TRIPHI and TRIPLO do not perform a signed comparison. It is recommended to leave OFFREF as 0 when using limit compare functionality.
- If different actions need to be taken for different PPB events from the same ADC module, then the ADCEVTINT ISR 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.

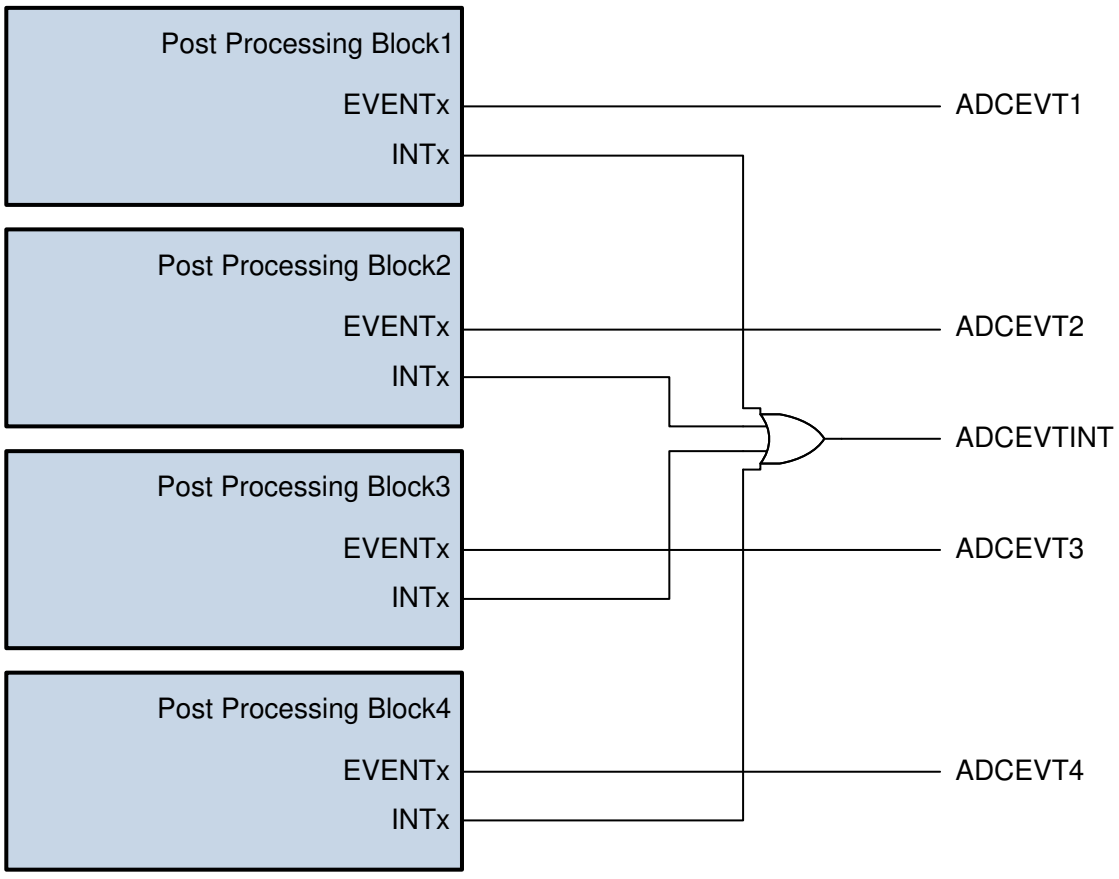



**Figure 11-10. ADC PPB Interrupt Event**



Copyright © 2024 Texas Instruments Incorporated

### 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



Copyright © 2024 Texas Instruments Incorporated

## 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 11-11.

**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.

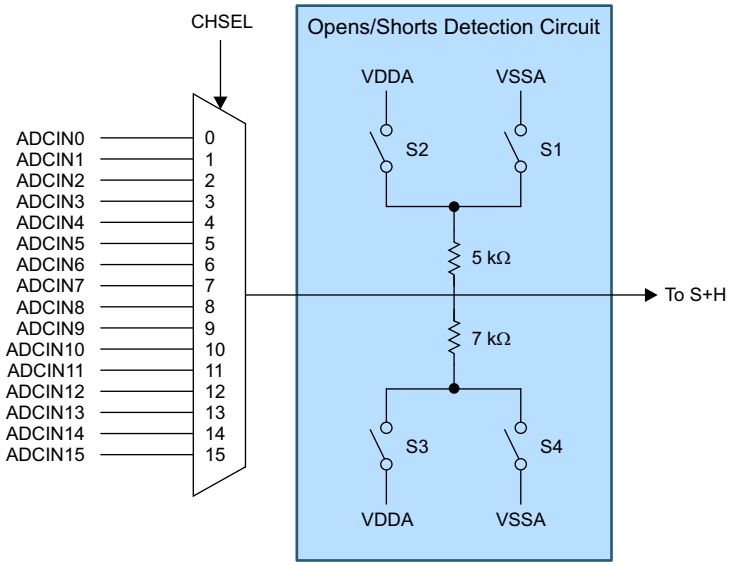



**Figure 11-11. Opens/Shorts Detection Circuit**

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 11-9.

**Table 11-9. 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 |



Copyright © 2024 Texas Instruments Incorporated

### 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 11-12 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 11-9 for the different configuration settings. Refer to Figure 11-12 when deriving the input signal to S/H if signal source $V_S$ is driving while the OSDETECT feature is enabled.

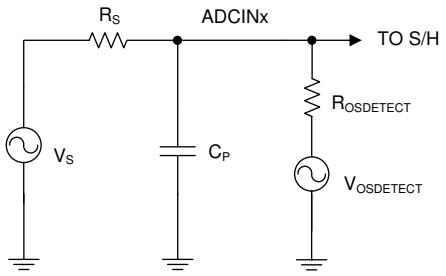



**Figure 11-12. Input Circuit Equivalent with OSDETECT Enabled**

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 11-9.
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 11-9.

### 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.



Copyright © 2024 Texas Instruments Incorporated

## 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 and the offset of the buffered DACs. 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 and DAC offset from OTP memory to the respective trim registers.
- The CalAdcINL() function copies the trim values for linearity from OTP memory to the respective trim registers for the specified ADC.
- A different offset trim is required for each possible combination of resolution and signal mode. The GetAdcOffsetTrimOTP(Uint16) function takes an input value corresponding to the ADC, resolution, and signal mode and returns the corresponding offset trim value from OTP memory, which the user then moves into the ADC offset trim register.
- The trim functions in Device_cal() are callable in C2000ware as ADC_setOFFSETTRIM(), ADC_setINLTRIM(), and DAC_setDACTRIM(). These functions fetch production test trim values from TI reserved OTP memory locations, and write the values to the corresponding analog module register destinations. Because these values are stored in the CPU1 Flash subsystem, the functions must be executed from CPU1 to fetch the correct values.

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).

**Note**

Trim and calibration values are stored in the OTP region of CPU1 Flash, and cannot be accessed by CPU2. When calling user calibration functions to update ADC, DAC or temperature sensor values, the function must be called from CPU1 code.

Refer to Section 11.11.2 for a tabulated description of associated function calls and ADC registers being populated during ADC calibration.



Copyright © 2024 Texas Instruments Incorporated

### 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, and external channels for differential operation.

Zero offset error is defined as the difference from 0 that occurs when converting a voltage at VREFLO (for single-ended operation), or the difference from MAX_CODE/2 when converting ADCINxP = ADCINxN (for differential mode). The zero offset error can be positive or negative. To correct this error, an adjustment of equal magnitude and opposite polarity is written into the ADCOFFTRIM register. The value contained in this register 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.

Using the GetAdcOffsetTrimOTP(Uint16) function, the ADCOFFTRIM register can be loaded with the factory calibrated offset error correction. The user can modify the ADCOFFTRIM register to compensate for additional offset error induced by the application environment if desired, but this is not typically necessary to achieve data manual specified performance.

**Note**

Regardless of the converter resolution, the size of each ADCOFFTRIM step is (VREFHI-VREFLO)/ 65536.

Use the following procedure to re-calibrate the ADC offset in 12-bit single-ended mode:

1. Set ADCOFFTRIM to +112 steps (0x70). This adds an artificial offset to account for negative offset that 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.

Use the following procedure to recalibrate the ADC offset in 16-bit differential mode:

1. Set ADCOFFTRIM to no adjustment (0x00).
2. Short ADCINxP and ADCINyN together (external connection) to a voltage near Vrefcm and accumulate some multiple of 16 conversions (for example, 32*16 conversions = 512 conversions).
3. Divide the accumulated result by the number of conversions (for example, for 512 conversions, divide by 512 ).
4. Set ADCOFFTRIM to 0 – result from step 3).



Copyright © 2024 Texas Instruments Incorporated

### 11.11.2 ADC Calibration Routines in OTP Memory

During factory testing, the calibration values for ADC offset (ADCOFFTRIM register) and ADC linearity correction (ADCINLTRIM1-6 registers) are measured per device. The resulting correction values for ADC offset and linearity and the associated functions to populate these into the corresponding ADC registers are stored in the OTP memory.Table 11-10 describes the function call, address pointer locations for the functions, and ADC registers being populated during device boot up sequence call to Device_cal() or any time the function call ADC_setMode() in C2000Ware is invoked. These functions generally must be called before using the ADC peripheral, and must be executed on CPU1. Following is an example of calibrating the ADC using C2000Ware driverlib functions. The ADC base macros used here come from within driverlib and are available for all ADC module instances.

```
//
// Calibrates the ADC reference, DAC offset, and internal oscillators
//
Device_cal();
//
// Calibrates the ADC A linearity; it is recommended to call the ADC_setMode function to
// properly calibrate the ADC linearity. The ADC_setINLTrim function will determine which
// linearity calibration function to call based on the ADC base that is passed in
//
ADC_setINLTrim(ADCA_BASE);
//
// Calibrates the offset trim for ADC A; it is recommended to call the ADC_setMode
// function to properly calibrate the ADC offset trim
//
ADC_setOffsetTrim(ADCA_BASE);
```

**Table 11-10. ADC Calibration Functions in OTP**

| Function Name | Pointer Address (OTP) | Description | Registers Populated |
|---|---|---|---|
| Device_cal() | 0x070282 | Calibration function for ADC reference, DAC offset, and internal oscillators | ANAREFTRIMA, ANAREFTRIMB, ANAREFTRIMC, ANAREFTRIMD, DACA OFFTRIM, DACB OFFTRIM, DACC OFFTRIM |
| ADC_calADCAINL() | 0x0703B4 | Calibration function for ADCA linearity; previously called by: `CalAdcaINL` | ADCA INLTRIM1-6 |
| ADC_calADCBINL() | 0x0703B2 | Calibration function for ADCB linearity; previously called by: `CalAdcbINL` | ADCB INLTRIM1-6 |
| ADC_calADCCINL() | 0x0703B0 | Calibration function for ADCC linearity; previously called by: `CalAdccINL` | ADCC INLTRIM1-6 |
| ADC_calADCDINL() | 0x0703AE | Calibration function for ADCD linearity; previously called by: `CalAdcdINL` | ADCD INLTRIM1-6 |
| ADC_getOffsetTrim() | 0x0703AC | Calibration function for ADCA-D offset; previously called by: `GetAdcOffsetTrimOTP` | ADCA-D ADCOFFTRIM |
| getTempSenseOffset() | 0x0703B8 | Function to read Temperature Sensor offset programmed at factory | none |
| getTempSenseSlope() | 0x0703B6 | Function to read Temperature Sensor slope programmed at factory | none |



Copyright © 2024 Texas Instruments Incorporated

## 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 in 12-bit mode and 29.5 ADCCLK cycles in 16-bit mode. The exact conversion time is always a whole number of SYSCLK cycles. See the timing diagrams and tables in Section 11.12.1 for exact timings.

### *11.12.1 ADC Timing Diagrams*

The following diagrams show the ADC conversion timings for two SOCs given the following assumptions:

- SOC0 and SOC1 are configured to use the same trigger.
- No other SOCs 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 11-11 describes the parameters in the following timing diagrams. Table 11-12 and Table 11-13 list the ADC timings..

**Table 11-11. ADC Timing Parameter Descriptions**

| Parameter | Description |
|---|---|
| $t_{SH}$ | The duration of the S+H window.<br>At the end of this window, the value on the S+H capacitor becomes the voltage to be converted into a digital value. The duration is given by (ACQPS + 1) SYSCLK cycles. ACQPS can be configured individually for each SOC, so $t_{SH}$ is not necessarily the same for different SOCs.<br>**Note:** The value on the S+H capacitor is captured approximately 5ns before the end of the S+H window regardless of device clock settings. |
| $t_{LAT}$ | The time from the end of the S+H window until the ADC results latch in the ADCRESULTx register.<br>If the ADCRESULTx register is read before this time, the previous conversion results are returned. |
| $t_{EOC}$ | The time from the end of the S+H window until the S+H window for the next ADC conversion can begin. In 16-bit mode, this coincides with the latching of the conversion results, while in 12-bit mode, the subsequent sample can start before the conversion results are latched. |
| $t_{INT}$ | The time from the end of the S+H window until an ADCINT flag is set (if configured).<br>If the INTPULSEPOS bit in the ADCCTL1 register is set, $t_{INT}$ coincides with the end of conversion (EOC) signal.<br>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.<br>If the INTPULSEPOS bit is 0, $t_{INT}$ coincides with the end of the S+H window. If $t_{INT}$ triggers a read of the ADC result register (directly through DMA or indirectly by triggering an ISR that reads the result), care must be taken to make sure the read occurs after the results latch (otherwise, the previous results are read). |



Copyright © 2024 Texas Instruments Incorporated

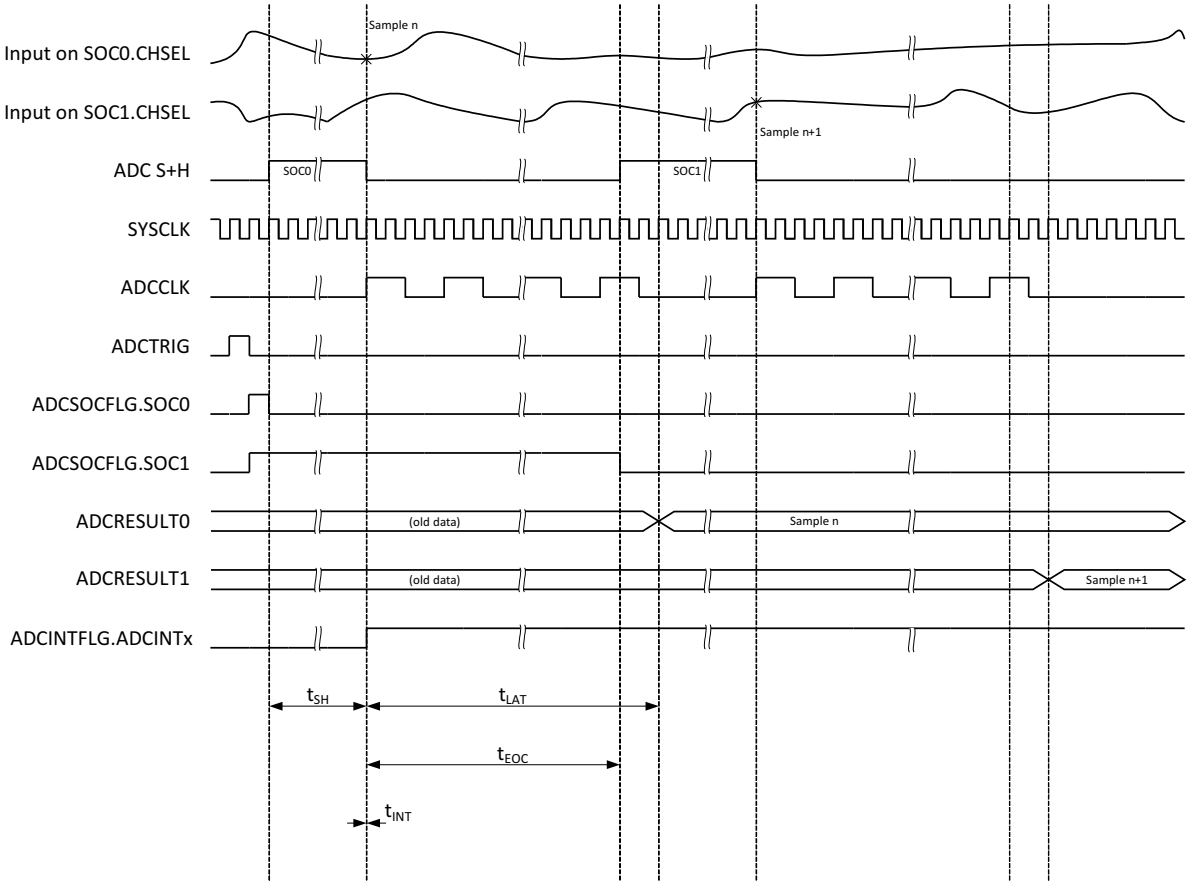



**Figure 11-13. ADC Timings for 12-bit Mode in Early Interrupt Mode**


Copyright © 2024 Texas Instruments Incorporated

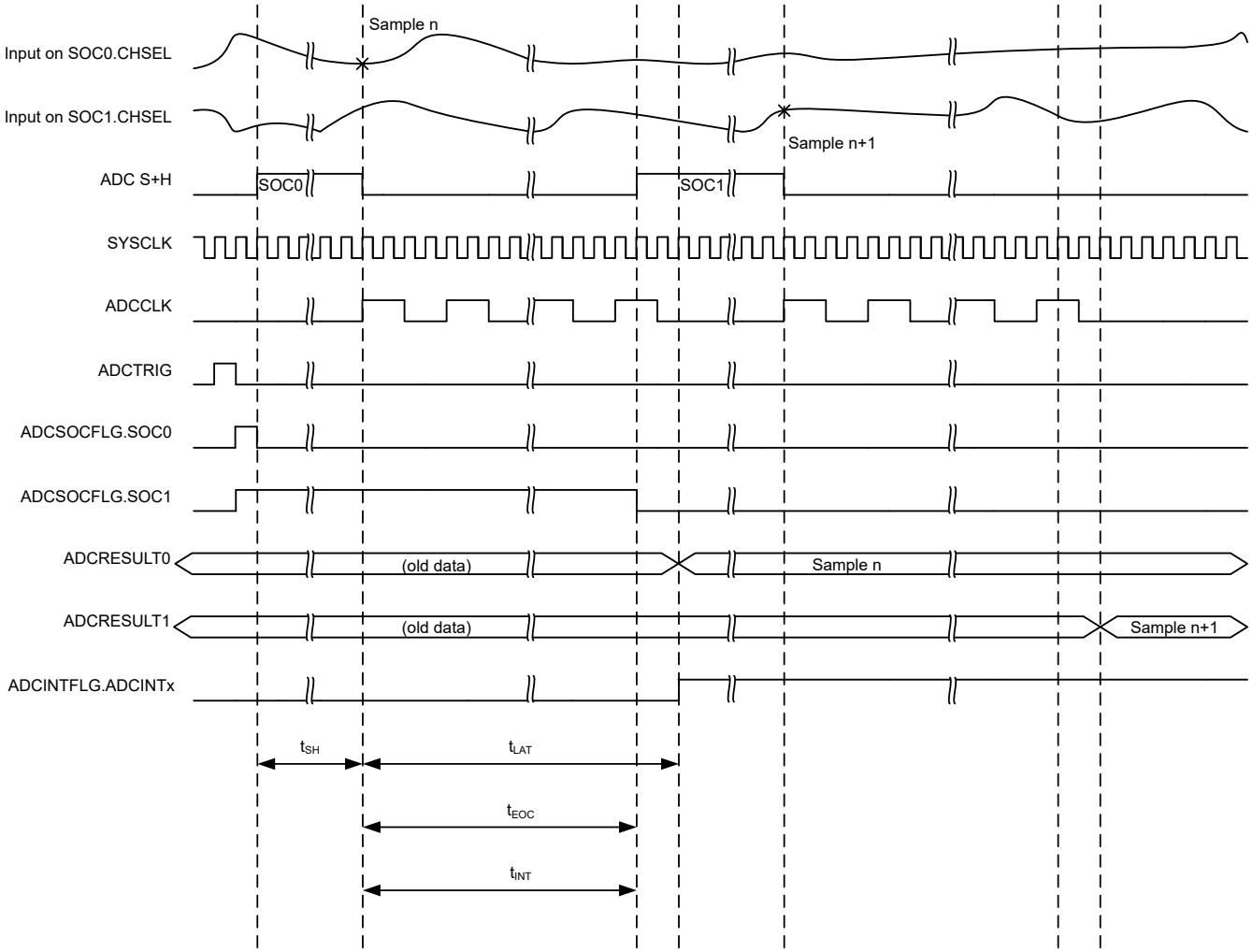



**Figure 11-14. ADC Timings for 12-bit Mode in Late Interrupt Mode**



Copyright © 2024 Texas Instruments Incorporated

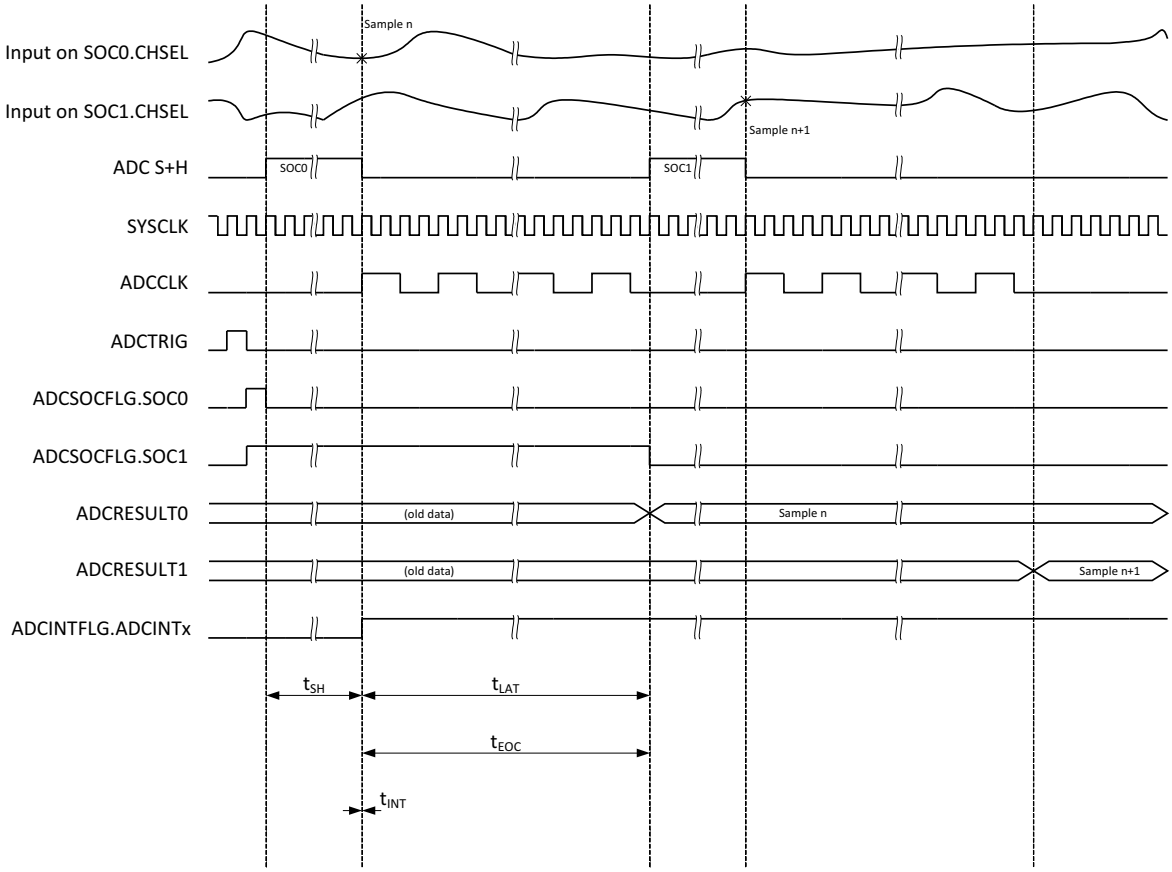



**Figure 11-15. ADC Timings for 16-bit Mode in Early Interrupt Mode**


Copyright © 2024 Texas Instruments Incorporated

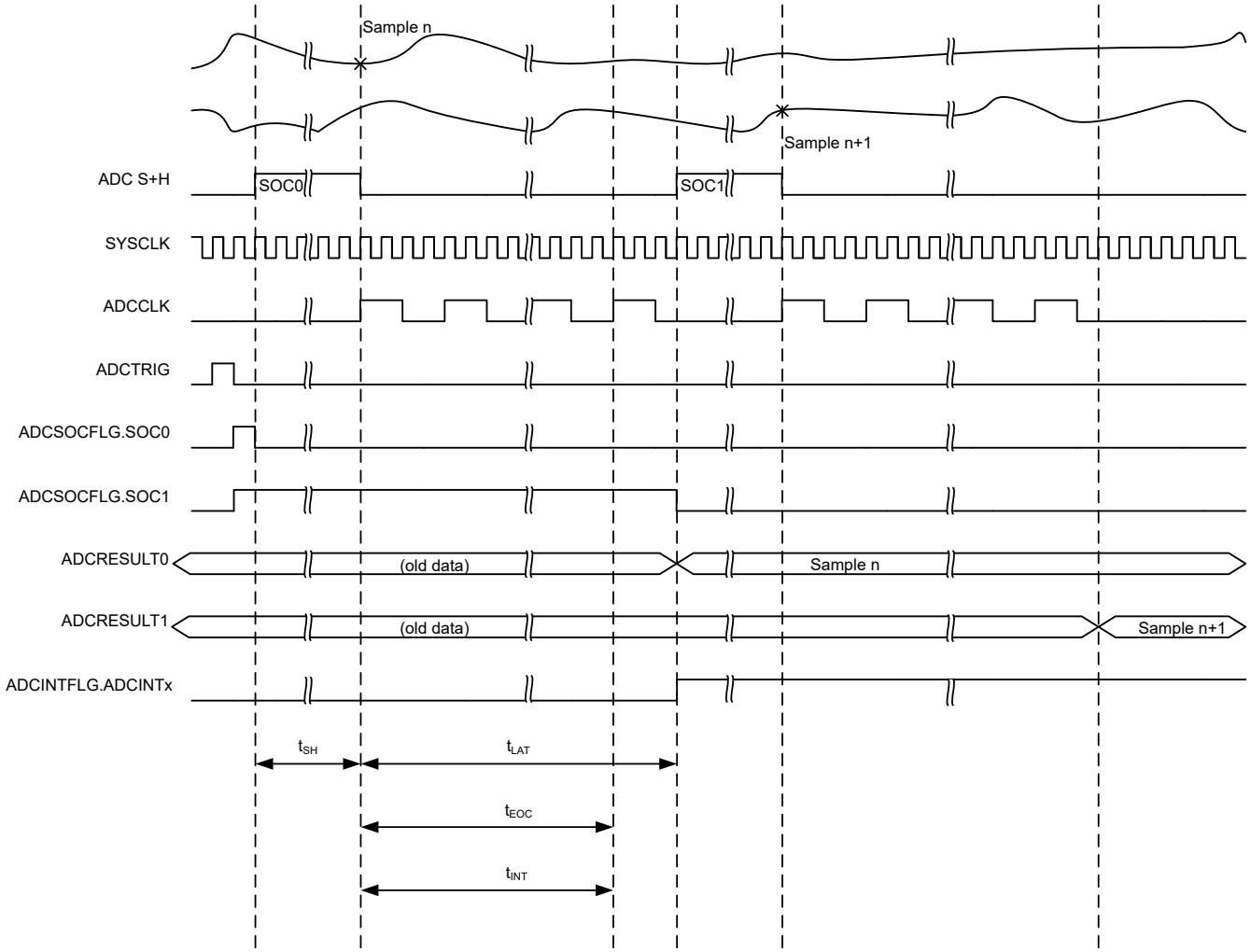



**Figure 11-16. ADC Timings for 16-bit Mode in Late Interrupt Mode (SYSCLK Cycles)**



Copyright © 2024 Texas Instruments Incorporated

**Table 11-12. ADC Timings in 12-bit Mode**

| ADCCLK Prescale | | SYSCLK Cycles | | | |
|---|---|---|---|---|---|
| ADCCTL2. PRESCALE | Prescale Ratio | $t_{EOC}$ | $t_{LAT}$ (1) | $t_{INT}$ (Early)(2) | $t_{INT}$ (Late) |
| 0 | 1 | 11 | 13 | 0 | 11 |
| 2 | 2 | 21 | 23 | 0 | 21 |
| 3 | 2.5 | 26 | 28 | 0 | 26 |
| 4 | 3 | 31 | 34 | 0 | 31 |
| 5 | 3.5 | 36 | 39 | 0 | 36 |
| 6 | 4 | 41 | 44 | 0 | 41 |
| 7 | 4.5 | 46 | 49 | 0 | 46 |
| 8 | 5 | 51 | 55 | 0 | 51 |
| 9 | 5.5 | 56 | 60 | 0 | 56 |
| 10 | 6 | 61 | 65 | 0 | 61 |
| 11 | 6.5 | 66 | 70 | 0 | 66 |
| 12 | 7 | 71 | 76 | 0 | 71 |
| 13 | 7.5 | 76 | 81 | 0 | 76 |
| 14 | 8 | 81 | 86 | 0 | 81 |
| 15 | 8.5 | 86 | 91 | 0 | 86 |

(1) Refer to the "ADC: DMA Read of Stale Result" advisory in the *TMS320F2837xD Dual-Core MCUs Silicon Errata*.
(2) By default, $t_{INT}$ occurs one SYSCLK cycle after the S+H window if INTPULSEPOS is 0.

**Table 11-13. ADC Timings in 16-bit Mode**

| ADCCLK Prescale | | SYSCLK Cycles | | | |
|---|---|---|---|---|---|
| ADCCTL2. PRESCALE | Prescale Ratio | $t_{EOC}$ | $t_{LAT}$ (1) | $t_{INT}$ (Early)(2) | $t_{INT}$ (Late) |
| 0 | 1 | 31 | 32 | 0 | 31 |
| 2 | 2 | 60 | 61 | 0 | 60 |
| 3 | 2.5 | 75 | 75 | 0 | 75 |
| 4 | 3 | 90 | 91 | 0 | 90 |
| 5 | 3.5 | 104 | 106 | 0 | 104 |
| 6 | 4 | 119 | 120 | 0 | 119 |
| 7 | 4.5 | 134 | 134 | 0 | 134 |
| 8 | 5 | 149 | 150 | 0 | 149 |
| 9 | 5.5 | 163 | 165 | 0 | 163 |
| 10 | 6 | 178 | 179 | 0 | 178 |
| 11 | 6.5 | 193 | 193 | 0 | 193 |
| 12 | 7 | 208 | 209 | 0 | 208 |
| 13 | 7.5 | 222 | 224 | 0 | 222 |
| 14 | 8 | 237 | 238 | 0 | 237 |
| 15 | 8.5 | 252 | 252 | 0 | 252 |

(1) Refer to the "ADC: DMA Read of Stale Result" advisory in the *TMS320F2837xD Dual-Core MCUs Silicon Errata*.
(2) By default, $t_{INT}$ occurs one SYSCLK cycle after the S+H window if INTPULSEPOS is 0.



Copyright © 2024 Texas Instruments Incorporated

## 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 SOCs each on ADCA and ADCB with identical trigger select and ACQPS values. This results in synchronous operation between ADCA and ADCB. For devices with more than two ADCs, the same principles can be used to synchronize all the ADCs.

```
Example: Basic Synchronous Operation
AdcaRegs.ADCSOC0CTL.bit.CHSEL = 4;      //SOC0 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
AdcbRegs.ADCSOC0CTL.bit.CHSEL = 0;      //SOC0 converts ADCINB0
AdcbRegs.ADCSOC0CTL.bit.ACQPS = 19;     //SOC0 uses sample duration of 20 SYSCLK cycles
AdcbRegs.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
AdcbRegs.ADCSOC1CTL.bit.CHSEL = 1;      //SOC1 converts ADCINB1
AdcbRegs.ADCSOC1CTL.bit.ACQPS = 30;     //SOC1 uses sample duration of 31 SYSCLK cycles
AdcbRegs.ADCSOC1CTL.bit.TRIGSEL = 10;   //SOC1 begins conversion on ePWM3 SOCB
```

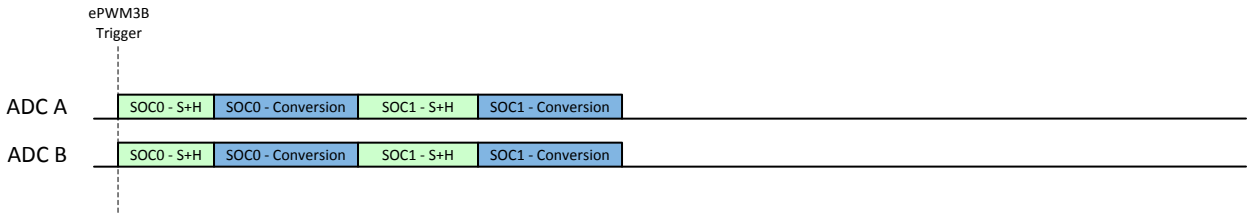



**Figure 11-17. Example: Basic Synchronous Operation**

Several things can be noted from Figure 11-17. First, while the ACQPS values must be the same for SOCs with the same number, different ACQPS values can be used for SOCs 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 SOCs are to be used, the priority must be configured the same on all ADCs.



Copyright © 2024 Texas Instruments Incorporated

#### 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 ADCB while using three SOCs and two trigger sources. Figure 11-18 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
AdcbRegs.ADCSOC0CTL.bit.CHSEL = 0;      //SOC0 converts ADCINB0
AdcbRegs.ADCSOC0CTL.bit.ACQPS = 19;     //SOC0 uses sample duration of 20 SYSCLK cycles
AdcbRegs.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
AdcbRegs.ADCSOC1CTL.bit.CHSEL = 1;      //SOC1 converts ADCINB1
AdcbRegs.ADCSOC1CTL.bit.ACQPS = 30;     //SOC1 uses sample duration of 31 SYSCLK cycles
AdcbRegs.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
AdcbRegs.ADCSOC2CTL.bit.CHSEL = 2;      //SOC2 converts ADCINB2
AdcbRegs.ADCSOC2CTL.bit.ACQPS = 19;     //SOC2 uses sample duration of 20 SYSCLK cycles
AdcbRegs.ADCSOC2CTL.bit.TRIGSEL = 2;    //SOC2 begins conversion on CPU Timer1
```

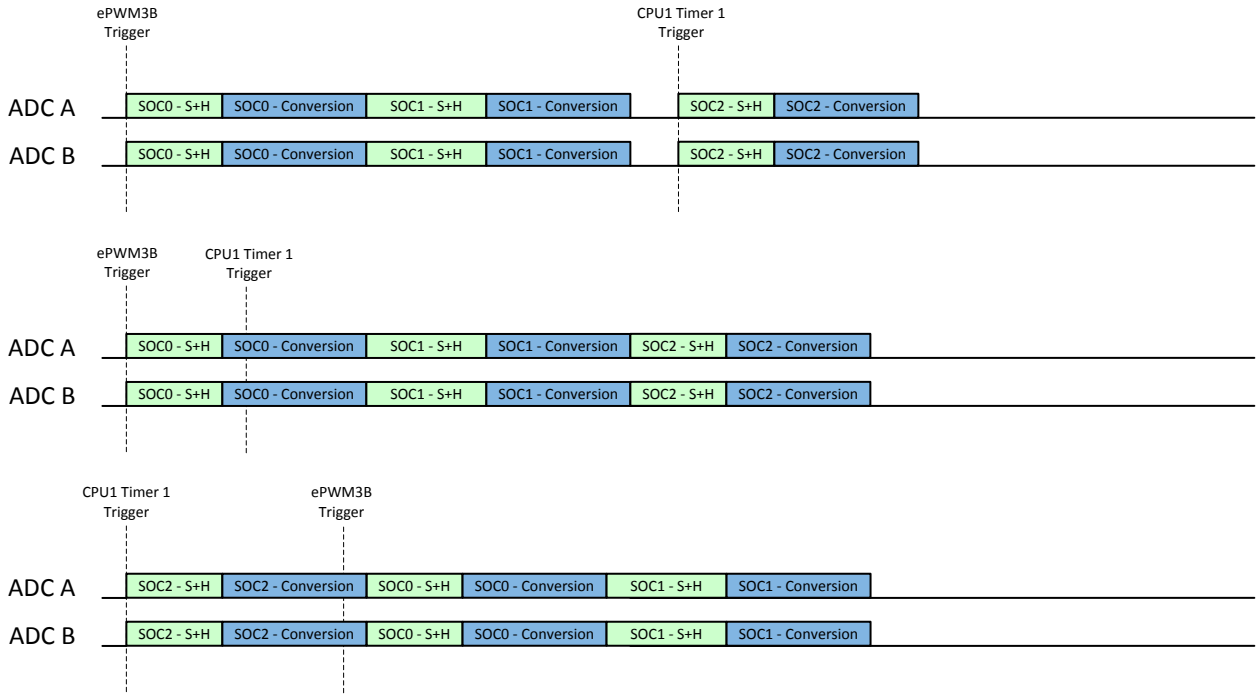



**Figure 11-18. Example: Synchronous Operation with Multiple Trigger Sources**

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.



Copyright © 2024 Texas Instruments Incorporated

#### 11.13.1.3 Synchronous Operation with Uneven SOC Numbers

If only one trigger source is used, one ADC can use more SOCs 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
AdcbRegs.ADCSOC0CTL.bit.CHSEL = 0;      //SOC0 converts ADCINB0
AdcbRegs.ADCSOC0CTL.bit.ACQPS = 19;     //SOC0 uses sample duration of 20 SYSCLK cycles
AdcbRegs.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
AdcbRegs.ADCSOC1CTL.bit.CHSEL = 1;      //SOC1 converts ADCINB1
AdcbRegs.ADCSOC1CTL.bit.ACQPS = 30;     //SOC1 uses sample duration of 31 SYSCLK cycles
AdcbRegs.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 = 10;   //SOC2 begins conversion on ePWM3 SOCB
```

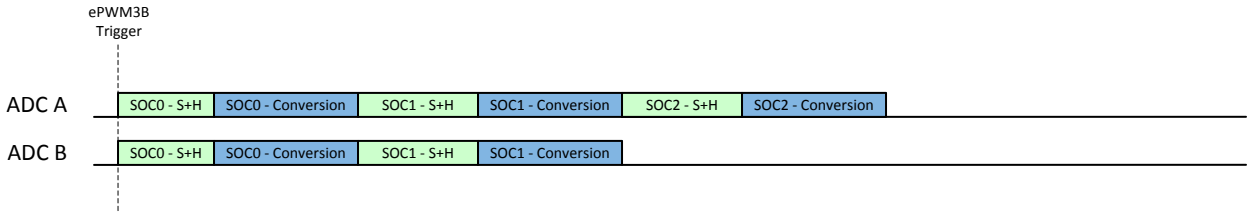



**Figure 11-19. Example: Synchronous Operation with Uneven SOC Numbers**

Note that if the trigger comes again before all SOCs have completed the conversions, ADCB 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.

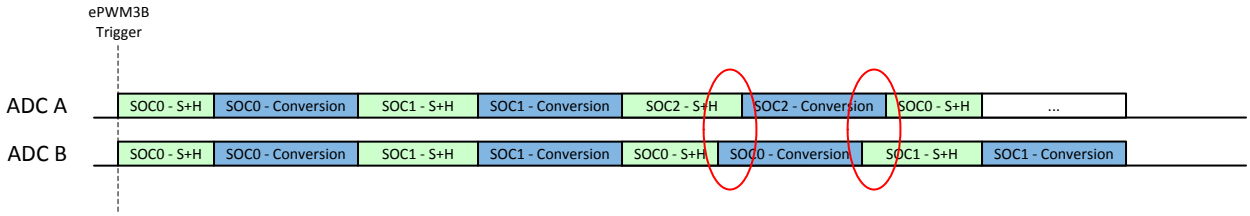



**Figure 11-20. Example: Asynchronous Operation with Uneven SOC Numbers – Trigger Overflow**



Copyright © 2024 Texas Instruments Incorporated

#### 11.13.1.4 Synchronous Operation with Different Resolutions

Configuring different ADCs to use different resolutions results in asynchronous operation. This occurs because the conversion time for 12-bit mode and 16-bit mode are different. Synchronous operation requires both the start and end of the conversion phase to be aligned, so even using the same S+H window duration does not result in synchronous operation.

```
Example: Asynchronous Operation with Different Resolutions
//ADCA = 12-bit mode
AdcaRegs.ADCSOC0CTL.bit.CHSEL = 4;      //SOC0 converts ADCINA4
AdcaRegs.ADCSOC0CTL.bit.ACQPS = 50;     //SOC0 uses sample duration of 51 SYSCLK cycles
AdcaRegs.ADCSOC0CTL.bit.TRIGSEL = 10;   //SOC0 begins conversion on ePWM3 SOCB

//ADCB = 16-bit mode
AdcbRegs.ADCSOC0CTL.bit.CHSEL = 0;      //SOC0 converts ADCINB0/B1
AdcbRegs.ADCSOC0CTL.bit.ACQPS = 50;     //SOC0 uses sample duration of 51 SYSCLK cycles
AdcbRegs.ADCSOC0CTL.bit.TRIGSEL = 10;   //SOC0 begins conversion on ePWM3 SOCB
```

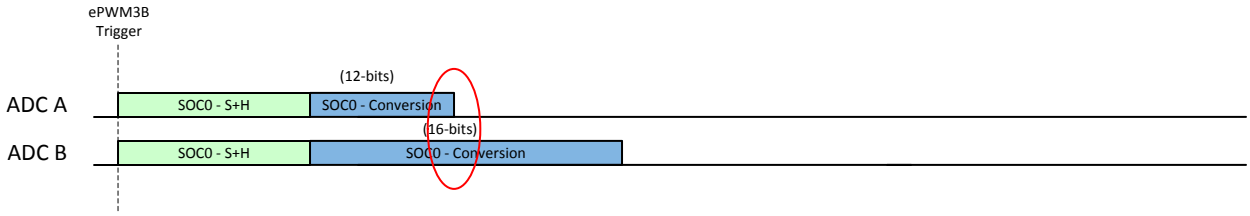



**Figure 11-21. Example: Asynchronous Operation with Different Resolutions**

To achieve synchronous operation while using both 12-bit and 16-bit resolution, conversions must be done in parallel at one resolution. Once conversions are complete at one resolution, the CPU must switch the resolution on all ADCs and then cause another trigger (this trigger must not be a software SOC force, as all ADCs cannot be started simultaneously using this method).

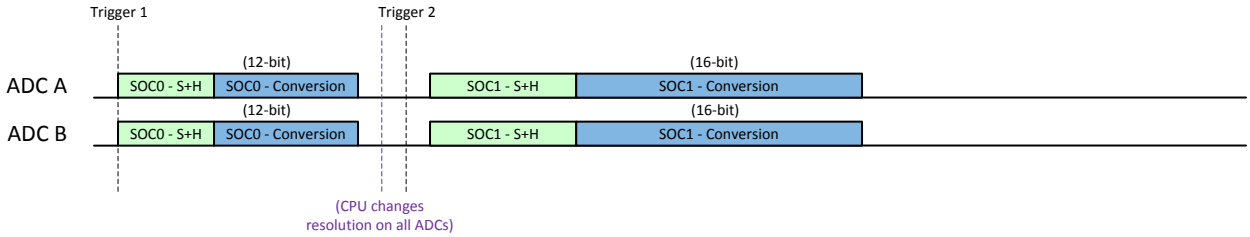



**Figure 11-22. Example: Synchronous Operation with Different Resolutions**



Copyright © 2024 Texas Instruments Incorporated

#### 11.13.1.5 Non-overlapping Conversions

If conversion timings can be made sure to not overlap by the user, then it is not necessary to configure all SOCs identically on all ADCs to achieve performance equivalent to synchronous operation. For example, if the two ADC triggers in a system come from two ePWM sources that are always 180-degrees out-of-phase, then SOC0 can be used for both ADCA and ADCB with different trigger sources and different ACQPS values.

```
Example: Operation with Non-overlapping Conversions
//ePWM3 SOCA and SOCB are 180 degrees out of phase
AdcaRegs.ADCSOC0CTL.bit.CHSEL = 4;      //SOC0 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
AdcbRegs.ADCSOC0CTL.bit.CHSEL = 0;      //SOC0 converts ADCINB0
AdcbRegs.ADCSOC0CTL.bit.ACQPS = 19;     //SOC0 uses sample duration of 20 SYSCLK cycles
AdcbRegs.ADCSOC0CTL.bit.TRIGSEL = 9;    //SOC0 begins conversion on ePWM3 SOCA
```

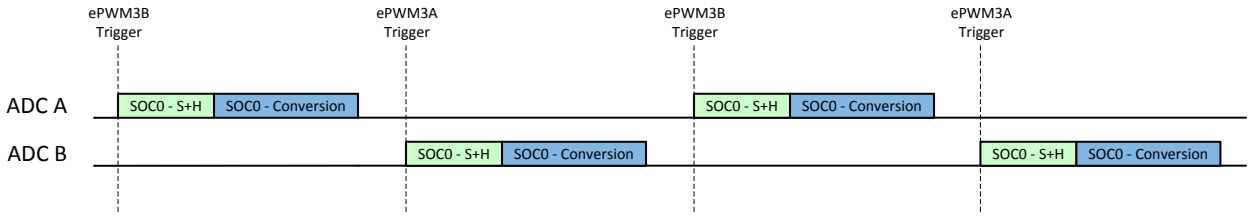



**Figure 11-23. Example: Synchronous Equivalent Operation with Non-Overlapping Conversions**

### *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:

$$\tau = (R_S + R_{on}) \times C_h + R_S \times \left(C_S + C_p\right) \tag{10}$$

And the number of time constants needed is given by the equation:

$$k = ln\left(\frac{2^n}{settling\ error}\right) - ln\left(\frac{C_S + C_P}{CH}\right) \tag{11}$$

So the total S+H time must be set to at least:

$$t = k \cdot \tau \tag{12}$$



Copyright © 2024 Texas Instruments Incorporated

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 Ω)
- $C_H$ = 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 Ω or kΩ)
- $C_S$ = capacitance on ADC input pin (typically in pF or nF)

For example, assuming the following parameters:

- n = 12-bits
- $R_{ON}$ = 500Ω
- $C_H$ = 12.5pF
- $C_p$ = 12.7pF
- settling error = ¼ LSB
- $R_s$ = 180Ω
- $C_s$ = 150pF

The time constant is calculated as:

$$\tau = (180\Omega + 500\Omega) \times 12.5pF + 180\Omega \times (150pF + 12.7pF) = 37.8ns \quad (13)$$

And the number of required time constants is:

$$k = ln\left(\frac{2^{12}}{0.25}\right) - ln\left(\frac{150pF + 12.7pF}{12.5pF}\right) = 9.70 - 2.57 = 7.13 \quad (14)$$

So the S+H time must be set to at least: 37.8ns × 7.13 = 270ns

If SYSCLK = 200MHz , then each SYSCLK cycle is 5ns. S+H duration is 270ns/5ns = 54 SYSCLK cycles, so ACQPS for this input is set to at least CEILING(54.0) – 1 = 53.

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.



Copyright © 2024 Texas Instruments Incorporated

### 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 4 ADCs based on an ePWM3 event. ADCINA3, ADCINB5, ADCINC5, and ADCIND2 are sampled. An acquisition window of 20 SYSCLK cycles is used, but different durations are possible.

```
AdcaRegs.ADCSOC0CTL.bit.CHSEL = 3;        //SOC0 convert 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
AdcbRegs.ADCSOC0CTL.bit.CHSEL = 5;        //SOC0 convert ADCINB5
AdcbRegs.ADCSOC0CTL.bit.ACQPS = 19;       //SOC0 uses sample duration of 20 SYSCLK cycles
AdcbRegs.ADCSOC0CTL.bit.TRIGSEL = 10;     //SOC0 begins conversion on ePWM3 SOCB
AdccRegs.ADCSOC0CTL.bit.CHSEL = 5;        //SOC0 convert 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
AdcdRegs.ADCSOC0CTL.bit.CHSEL = 2;        //SOC0 convert ADCIND2
AdcdRegs.ADCSOC0CTL.bit.ACQPS = 19;       //SOC0 uses sample duration of 20 SYSCLK cycles
AdcdRegs.ADCSOC0CTL.bit.TRIGSEL = 10;     //SOC0 begins conversion on ePWM3 SOCB
```

When the ePWM3 trigger is received, all four 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, DMAs, and CLAs 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 ADCIN13 on ADCA by setting the ENABLE bit in the TSNSCTL register.

To convert the temperature sensor reading into a temperature, pass the temperature sensor reading to the GetTemperatureC() function in F2837xD_TempSensorConv.c .

Note that this function assumes that the temperature reading is taken with VREFHI = 2.5V. If a different reference voltage is used, the sample can be scaled appropriately before passing the sample to the function by using the following formula:

adjusted sensor reading = raw sensor reading * (VREFHI / 2.5V)

**Note**

To sample the temperature sensor, the ADC must be in single-ended 12-bit mode.

If the temperature sensor is sampled in 16-bit mode, the internal temperature sensor is sampled with a 12-bit resolution, regardless of the ADC mode currently configured. This does not change the resolution of results obtained on other SOCs for the given ADC.



Copyright © 2024 Texas Instruments Incorporated

### 11.13.6 Designing an External Reference Circuit

Figure 11-24 shows the basic organization of the external voltage reference generation circuitry. TI recommends that a single reference voltage generation source is shared by all ADC modules. This minimizes reference voltage mismatch between ADC modules. 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 <1Ω) in series with this capacitor is sometimes necessary to maintain op-amp stability.

To share two reference pins between one op-amp driver is possible. This organization is shown in Figure 11-25. This can give slightly reduced performance compared to the case where each reference pin has a dedicated op-amp buffer, but can still be possible to achieve all ADC specifications in the data sheet.

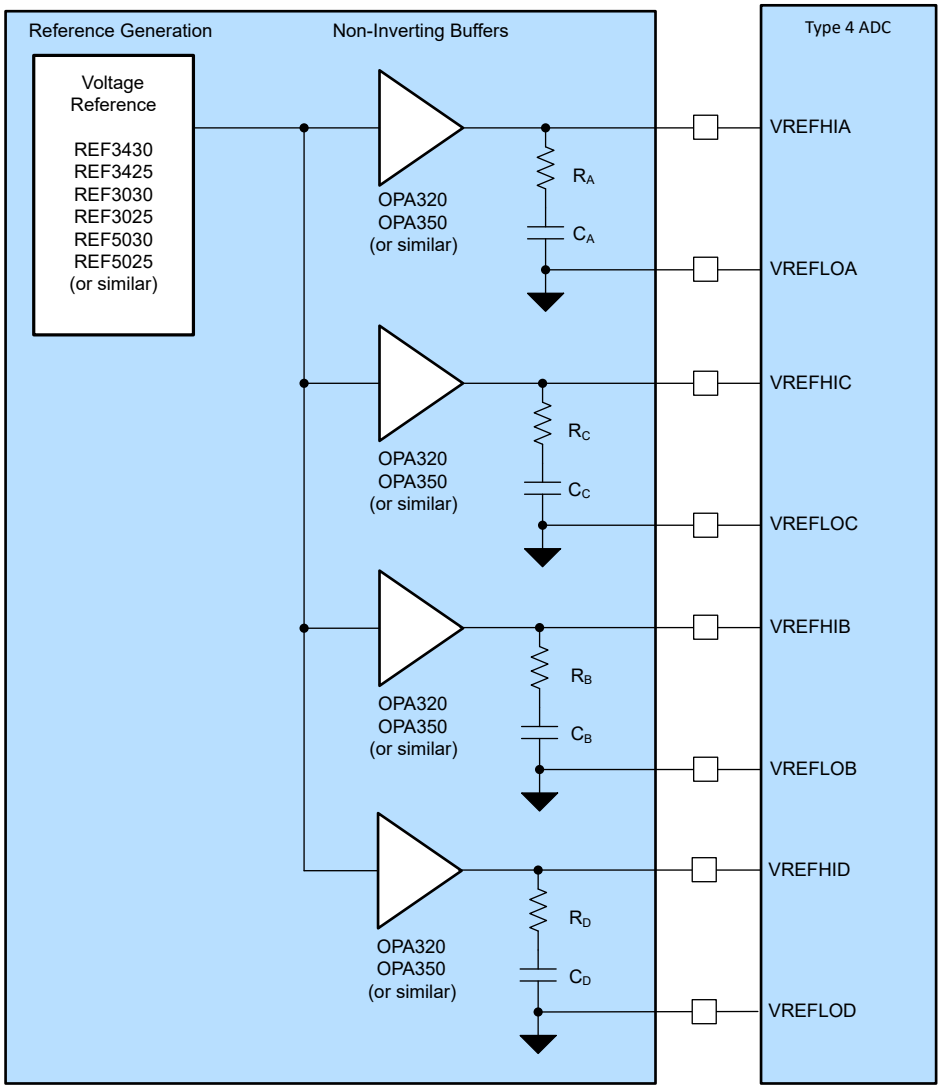



**Figure 11-24. ADC Reference System**



Copyright © 2024 Texas Instruments Incorporated

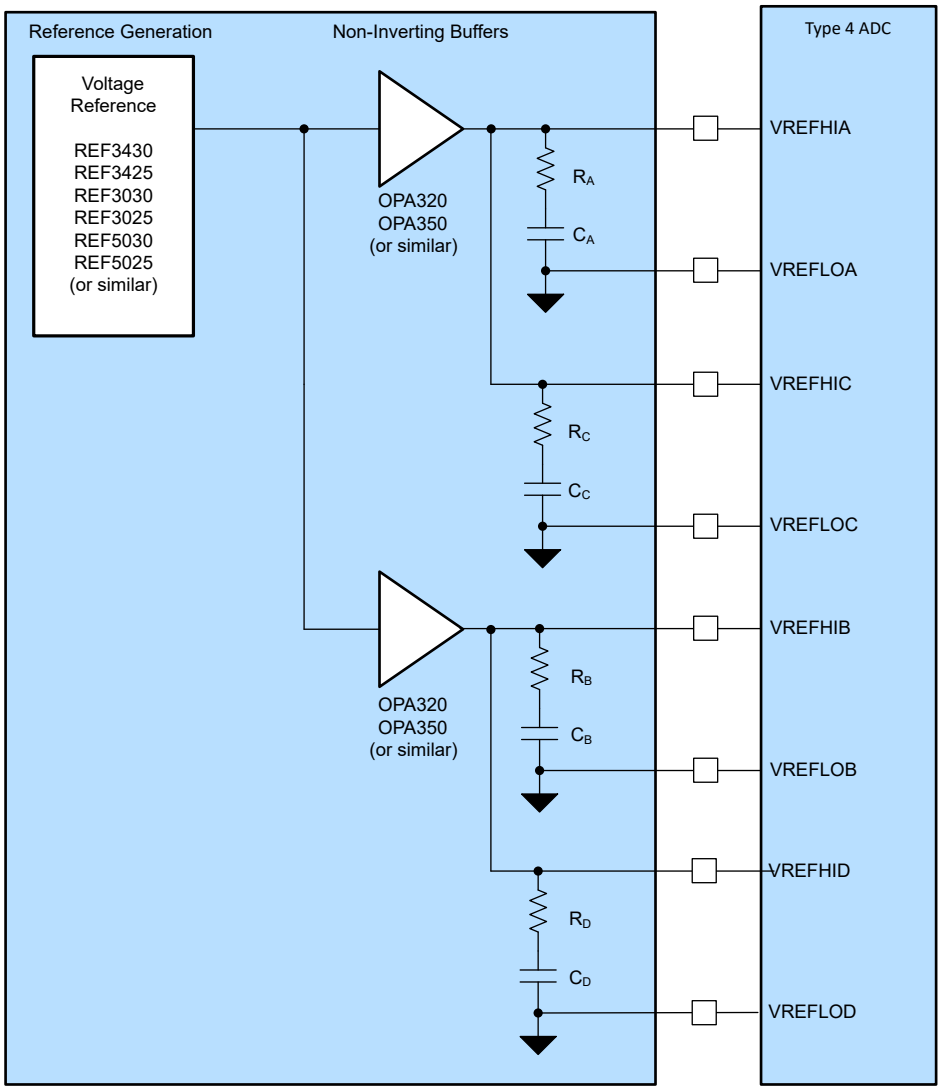



**Figure 11-25. ADC Shared Reference System**



Copyright © 2024 Texas Instruments Incorporated

## 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.

#### 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*

*Watch Variables*
- *myADC0Result0* - Digital representation of the voltage on pin A0
- *myADC0Result1* - Digital representation of the voltage on pin A1

#### 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*

*Watch Variables*
- *myADC0Result0* : a digital representation of the voltage on pin A2
- *myADC0Result1* : a digital representation of the voltage on pin A3

#### 11.14.1.5 ADC Continuous Triggering (adc_soc_continuous)

FILE: adc_ex5_soc_continuous.c



Copyright © 2024 Texas Instruments Incorporated

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*

*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

#### 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 of a batch of conversions using the available parallelism across multiple ADCs.

ADCA Interrupt ISRs are used to read results of both ADCA and ADCC.



Copyright © 2024 Texas Instruments Incorporated

*External Connections*

*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
- *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



Copyright © 2024 Texas Instruments Incorporated

#### 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
- 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

## 11.15 ADC Registers

This section describes the Analog-to-Digital Converter Registers.

### *11.15.1 ADC Base Addresses*

**Table 11-14. ADC Base Address Table**

| Device Registers | Register Name | Start Address | End Address |
|---|---|---|---|
| AdcaResultRegs | ADC_RESULT_REGS | 0x0000_0B00 | 0x0000_0B1F |
| AdcbResultRegs | ADC_RESULT_REGS | 0x0000_0B20 | 0x0000_0B3F |
| AdccResultRegs | ADC_RESULT_REGS | 0x0000_0B40 | 0x0000_0B5F |
| AdcdResultRegs | ADC_RESULT_REGS | 0x0000_0B60 | 0x0000_0B7F |
| AdcaRegs | ADC_REGS | 0x0000_7400 | 0x0000_747F |
| AdcbRegs | ADC_REGS | 0x0000_7480 | 0x0000_74FF |
| AdccRegs | ADC_REGS | 0x0000_7500 | 0x0000_757F |
| AdcdRegs | ADC_REGS | 0x0000_7580 | 0x0000_75FF |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.2 ADC_RESULT_REGS Registers

Table 11-15 lists the memory-mapped registers for the ADC_RESULT_REGS registers. All register offset addresses not listed in Table 11-15 should be considered as reserved locations and the register contents should not be modified.

**Table 11-15. ADC_RESULT_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | ADCRESULT0 | ADC Result 0 Register | | Go |
| 1h | ADCRESULT1 | ADC Result 1 Register | | Go |
| 2h | ADCRESULT2 | ADC Result 2 Register | | Go |
| 3h | ADCRESULT3 | ADC Result 3 Register | | Go |
| 4h | ADCRESULT4 | ADC Result 4 Register | | Go |
| 5h | ADCRESULT5 | ADC Result 5 Register | | Go |
| 6h | ADCRESULT6 | ADC Result 6 Register | | Go |
| 7h | ADCRESULT7 | ADC Result 7 Register | | Go |
| 8h | ADCRESULT8 | ADC Result 8 Register | | Go |
| 9h | ADCRESULT9 | ADC Result 9 Register | | Go |
| Ah | ADCRESULT10 | ADC Result 10 Register | | Go |
| Bh | ADCRESULT11 | ADC Result 11 Register | | Go |
| Ch | ADCRESULT12 | ADC Result 12 Register | | Go |
| Dh | ADCRESULT13 | ADC Result 13 Register | | Go |
| Eh | ADCRESULT14 | ADC Result 14 Register | | Go |
| Fh | ADCRESULT15 | ADC Result 15 Register | | Go |
| 10h | ADCPPB1RESULT | ADC Post Processing Block 1 Result Register | | Go |
| 12h | ADCPPB2RESULT | ADC Post Processing Block 2 Result Register | | Go |
| 14h | ADCPPB3RESULT | ADC Post Processing Block 3 Result Register | | Go |
| 16h | ADCPPB4RESULT | ADC Post Processing Block 4 Result Register | | Go |

Complex bit access types are encoded to fit into small table cells. Table 11-16 shows the codes that are used for access types in this section.

**Table 11-16. 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. |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.2.1 ADCRESULT0 Register (Offset = 0h) [Reset = 0000h]

ADCRESULT0 is shown in Figure 11-26 and described in Table 11-17.

Return to the Summary Table.

ADC Result 0 Register

**Figure 11-26. 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 11-17. ADCRESULT0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RESULT | R | 0h | ADC Result 0<br>16-bit ADC result. After the ADC completes a conversion of SOC0, the digital result is placed in this bit field.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.2.2 ADCRESULT1 Register (Offset = 1h) [Reset = 0000h]

ADCRESULT1 is shown in Figure 11-27 and described in Table 11-18.

Return to the Summary Table.

ADC Result 1 Register

**Figure 11-27. 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 11-18. ADCRESULT1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RESULT | R | 0h | ADC Result 1<br>16-bit ADC result. After the ADC completes a conversion of SOC1, the digital result is placed in this bit field.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.2.3 ADCRESULT2 Register (Offset = 2h) [Reset = 0000h]

ADCRESULT2 is shown in Figure 11-28 and described in Table 11-19.

Return to the Summary Table.

ADC Result 2 Register

**Figure 11-28. 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 11-19. ADCRESULT2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RESULT | R | 0h | ADC Result 2<br>16-bit ADC result. After the ADC completes a conversion of SOC2, the digital result is placed in this bit field.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.2.4 ADCRESULT3 Register (Offset = 3h) [Reset = 0000h]

ADCRESULT3 is shown in Figure 11-29 and described in Table 11-20.

Return to the Summary Table.

ADC Result 3 Register

**Figure 11-29. 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 11-20. ADCRESULT3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RESULT | R | 0h | ADC Result 3<br>16-bit ADC result. After the ADC completes a conversion of SOC3, the digital result is placed in this bit field.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.2.5 ADCRESULT4 Register (Offset = 4h) [Reset = 0000h]

ADCRESULT4 is shown in Figure 11-30 and described in Table 11-21.

Return to the Summary Table.

ADC Result 4 Register

**Figure 11-30. 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 11-21. ADCRESULT4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RESULT | R | 0h | ADC Result 4<br>16-bit ADC result. After the ADC completes a conversion of SOC4, the digital result is placed in this bit field.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.2.6 ADCRESULT5 Register (Offset = 5h) [Reset = 0000h]

ADCRESULT5 is shown in Figure 11-31 and described in Table 11-22.

Return to the Summary Table.

ADC Result 5 Register

**Figure 11-31. 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 11-22. ADCRESULT5 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RESULT | R | 0h | ADC Result 5<br>16-bit ADC result. After the ADC completes a conversion of SOC5, the digital result is placed in this bit field.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.2.7 ADCRESULT6 Register (Offset = 6h) [Reset = 0000h]

ADCRESULT6 is shown in Figure 11-32 and described in Table 11-23.

Return to the Summary Table.

ADC Result 6 Register

**Figure 11-32. 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 11-23. ADCRESULT6 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RESULT | R | 0h | ADC Result 6<br>16-bit ADC result. After the ADC completes a conversion of SOC6, the digital result is placed in this bit field.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.2.8 ADCRESULT7 Register (Offset = 7h) [Reset = 0000h]

ADCRESULT7 is shown in Figure 11-33 and described in Table 11-24.

Return to the Summary Table.

ADC Result 7 Register

**Figure 11-33. 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 11-24. ADCRESULT7 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RESULT | R | 0h | ADC Result 7<br>16-bit ADC result. After the ADC completes a conversion of SOC7, the digital result is placed in this bit field.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.2.9 ADCRESULT8 Register (Offset = 8h) [Reset = 0000h]

ADCRESULT8 is shown in Figure 11-34 and described in Table 11-25.

Return to the Summary Table.

ADC Result 8 Register

**Figure 11-34. 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 11-25. ADCRESULT8 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RESULT | R | 0h | ADC Result 8<br>16-bit ADC result. After the ADC completes a conversion of SOC8, the digital result is placed in this bit field.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.2.10 ADCRESULT9 Register (Offset = 9h) [Reset = 0000h]

ADCRESULT9 is shown in Figure 11-35 and described in Table 11-26.

Return to the Summary Table.

ADC Result 9 Register

**Figure 11-35. 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 11-26. ADCRESULT9 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RESULT | R | 0h | ADC Result 9<br>16-bit ADC result. After the ADC completes a conversion of SOC9, the digital result is placed in this bit field.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.2.11 ADCRESULT10 Register (Offset = Ah) [Reset = 0000h]

ADCRESULT10 is shown in Figure 11-36 and described in Table 11-27.

Return to the Summary Table.

ADC Result 10 Register

**Figure 11-36. 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 11-27. ADCRESULT10 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RESULT | R | 0h | ADC Result 10<br>16-bit ADC result. After the ADC completes a conversion of SOC10, the digital result is placed in this bit field.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.2.12 ADCRESULT11 Register (Offset = Bh) [Reset = 0000h]

ADCRESULT11 is shown in Figure 11-37 and described in Table 11-28.

Return to the Summary Table.

ADC Result 11 Register

**Figure 11-37. 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 11-28. ADCRESULT11 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RESULT | R | 0h | ADC Result 11<br>16-bit ADC result. After the ADC completes a conversion of SOC11, the digital result is placed in this bit field.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.2.13 ADCRESULT12 Register (Offset = Ch) [Reset = 0000h]

ADCRESULT12 is shown in Figure 11-38 and described in Table 11-29.

Return to the Summary Table.

ADC Result 12 Register

**Figure 11-38. 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 11-29. ADCRESULT12 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RESULT | R | 0h | ADC Result 12<br>16-bit ADC result. After the ADC completes a conversion of SOC12, the digital result is placed in this bit field.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.2.14 ADCRESULT13 Register (Offset = Dh) [Reset = 0000h]

ADCRESULT13 is shown in Figure 11-39 and described in Table 11-30.

Return to the Summary Table.

ADC Result 13 Register

**Figure 11-39. 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 11-30. ADCRESULT13 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RESULT | R | 0h | ADC Result 13<br>16-bit ADC result. After the ADC completes a conversion of SOC13, the digital result is placed in this bit field.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.2.15 ADCRESULT14 Register (Offset = Eh) [Reset = 0000h]

ADCRESULT14 is shown in Figure 11-40 and described in Table 11-31.

Return to the Summary Table.

ADC Result 14 Register

**Figure 11-40. 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 11-31. ADCRESULT14 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RESULT | R | 0h | ADC Result 14<br>16-bit ADC result. After the ADC completes a conversion of SOC14, the digital result is placed in this bit field.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.2.16 ADCRESULT15 Register (Offset = Fh) [Reset = 0000h]

ADCRESULT15 is shown in Figure 11-41 and described in Table 11-32.

Return to the Summary Table.

ADC Result 15 Register

**Figure 11-41. 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 11-32. ADCRESULT15 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RESULT | R | 0h | ADC Result 15<br>16-bit ADC result. After the ADC completes a conversion of SOC15, the digital result is placed in this bit field.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.2.17 ADCPPB1RESULT Register (Offset = 10h) [Reset = 00000000h]

ADCPPB1RESULT is shown in Figure 11-42 and described in Table 11-33.

Return to the Summary Table.

ADC Post Processing Block 1 Result Register

**Figure 11-42. 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 11-33. 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.<br>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.<br>Reset type: SYSRSn |
| 15-0 | PPBRESULT | R | 0h | ADC Post Processing Block Result 1<br>The result of the offset/reference subtraction post conversion processing is stored in this register.<br>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.<br>NOTE: If the conversion associated with this Post Processing Block is a 12-bit conversion, the PPBRESULT bits are limited to bits 12:0.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.2.18 ADCPPB2RESULT Register (Offset = 12h) [Reset = 00000000h]

ADCPPB2RESULT is shown in Figure 11-43 and described in Table 11-34.

Return to the Summary Table.

ADC Post Processing Block 2 Result Register

**Figure 11-43. 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 11-34. 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.<br>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.<br>Reset type: SYSRSn |
| 15-0 | PPBRESULT | R | 0h | ADC Post Processing Block Result 2<br>The result of the offset/reference subtraction post conversion processing is stored in this register.<br>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.<br>NOTE: If the conversion associated with this Post Processing Block is a 12-bit conversion, the PPBRESULT bits are limited to bits 12:0.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.2.19 ADCPPB3RESULT Register (Offset = 14h) [Reset = 00000000h]

ADCPPB3RESULT is shown in Figure 11-44 and described in Table 11-35.

Return to the Summary Table.

ADC Post Processing Block 3 Result Register

**Figure 11-44. 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 11-35. 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.<br>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.<br>Reset type: SYSRSn |
| 15-0 | PPBRESULT | R | 0h | ADC Post Processing Block Result 3<br>The result of the offset/reference subtraction post conversion processing is stored in this register.<br>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.<br>NOTE: If the conversion associated with this Post Processing Block is a 12-bit conversion, the PPBRESULT bits are limited to bits 12:0.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.2.20 ADCPPB4RESULT Register (Offset = 16h) [Reset = 00000000h]

ADCPPB4RESULT is shown in Figure 11-45 and described in Table 11-36.

Return to the Summary Table.

ADC Post Processing Block 4 Result Register

**Figure 11-45. 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 11-36. 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.<br>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.<br>Reset type: SYSRSn |
| 15-0 | PPBRESULT | R | 0h | ADC Post Processing Block Result 4<br>The result of the offset/reference subtraction post conversion processing is stored in this register.<br>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.<br>NOTE: If the conversion associated with this Post Processing Block is a 12-bit conversion, the PPBRESULT bits are limited to bits 12:0.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3 ADC_REGS Registers

Table 11-37 lists the memory-mapped registers for the ADC_REGS registers. All register offset addresses not listed in Table 11-37 should be considered as reserved locations and the register contents should not be modified.

**Table 11-37. ADC_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | ADCCTL1 | ADC Control 1 Register | EALLOW | Go |
| 1h | ADCCTL2 | ADC Control 2 Register | EALLOW | Go |
| 2h | ADCBURSTCTL | ADC Burst Control Register | EALLOW | Go |
| 3h | ADCINTFLG | ADC Interrupt Flag Register | | Go |
| 4h | ADCINTFLGCLR | ADC Interrupt Flag Clear Register | | Go |
| 5h | ADCINTOVF | ADC Interrupt Overflow Register | | Go |
| 6h | ADCINTOVFCLR | ADC Interrupt Overflow Clear Register | | Go |
| 7h | ADCINTSEL1N2 | ADC Interrupt 1 and 2 Selection Register | EALLOW | Go |
| 8h | ADCINTSEL3N4 | ADC Interrupt 3 and 4 Selection Register | EALLOW | Go |
| 9h | ADCSOCPRICTL | ADC SOC Priority Control Register | EALLOW | Go |
| Ah | ADCINTSOCSEL1 | ADC Interrupt SOC Selection 1 Register | EALLOW | Go |
| Bh | ADCINTSOCSEL2 | ADC Interrupt SOC Selection 2 Register | EALLOW | Go |
| Ch | ADCSOCFLG1 | ADC SOC Flag 1 Register | | Go |
| Dh | ADCSOCFRC1 | ADC SOC Force 1 Register | | Go |
| Eh | ADCSOCOVF1 | ADC SOC Overflow 1 Register | | Go |
| Fh | ADCSOCOVFCLR1 | ADC SOC Overflow Clear 1 Register | | Go |
| 10h | ADCSOC0CTL | ADC SOC0 Control Register | EALLOW | Go |
| 12h | ADCSOC1CTL | ADC SOC1 Control Register | EALLOW | Go |
| 14h | ADCSOC2CTL | ADC SOC2 Control Register | EALLOW | Go |
| 16h | ADCSOC3CTL | ADC SOC3 Control Register | EALLOW | Go |
| 18h | ADCSOC4CTL | ADC SOC4 Control Register | EALLOW | Go |
| 1Ah | ADCSOC5CTL | ADC SOC5 Control Register | EALLOW | Go |
| 1Ch | ADCSOC6CTL | ADC SOC6 Control Register | EALLOW | Go |
| 1Eh | ADCSOC7CTL | ADC SOC7 Control Register | EALLOW | Go |
| 20h | ADCSOC8CTL | ADC SOC8 Control Register | EALLOW | Go |
| 22h | ADCSOC9CTL | ADC SOC9 Control Register | EALLOW | Go |
| 24h | ADCSOC10CTL | ADC SOC10 Control Register | EALLOW | Go |
| 26h | ADCSOC11CTL | ADC SOC11 Control Register | EALLOW | Go |
| 28h | ADCSOC12CTL | ADC SOC12 Control Register | EALLOW | Go |
| 2Ah | ADCSOC13CTL | ADC SOC13 Control Register | EALLOW | Go |
| 2Ch | ADCSOC14CTL | ADC SOC14 Control Register | EALLOW | Go |
| 2Eh | ADCSOC15CTL | ADC SOC15 Control Register | EALLOW | Go |
| 30h | ADCEVTSTAT | ADC Event Status Register | | Go |
| 32h | ADCEVTCLR | ADC Event Clear Register | | Go |
| 34h | ADCEVTSEL | ADC Event Selection Register | EALLOW | Go |
| 36h | ADCEVTINTSEL | ADC Event Interrupt Selection Register | EALLOW | Go |
| 38h | ADCOSDETECT | ADC Open and Shorts Detect Register | EALLOW | Go |
| 39h | ADCCOUNTER | ADC Counter Register | | Go |
| 3Ah | ADCREV | ADC Revision Register | | Go |
| 3Bh | ADCOFFTRIM | ADC Offset Trim Register | EALLOW | Go |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-37. ADC_REGS Registers (continued)**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 40h | ADCPPB1CONFIG | ADC PPB1 Config Register | EALLOW | Go |
| 41h | ADCPPB1STAMP | ADC PPB1 Sample Delay Time Stamp Register | | Go |
| 42h | ADCPPB1OFFCAL | ADC PPB1 Offset Calibration Register | EALLOW | Go |
| 43h | ADCPPB1OFFREF | ADC PPB1 Offset Reference Register | | Go |
| 44h | ADCPPB1TRIPHI | ADC PPB1 Trip High Register | EALLOW | Go |
| 46h | ADCPPB1TRIPLO | ADC PPB1 Trip Low/Trigger Time Stamp Register | EALLOW | Go |
| 48h | ADCPPB2CONFIG | ADC PPB2 Config Register | EALLOW | Go |
| 49h | ADCPPB2STAMP | ADC PPB2 Sample Delay Time Stamp Register | | Go |
| 4Ah | ADCPPB2OFFCAL | ADC PPB2 Offset Calibration Register | EALLOW | Go |
| 4Bh | ADCPPB2OFFREF | ADC PPB2 Offset Reference Register | | Go |
| 4Ch | ADCPPB2TRIPHI | ADC PPB2 Trip High Register | EALLOW | Go |
| 4Eh | ADCPPB2TRIPLO | ADC PPB2 Trip Low/Trigger Time Stamp Register | EALLOW | Go |
| 50h | ADCPPB3CONFIG | ADC PPB3 Config Register | EALLOW | Go |
| 51h | ADCPPB3STAMP | ADC PPB3 Sample Delay Time Stamp Register | | Go |
| 52h | ADCPPB3OFFCAL | ADC PPB3 Offset Calibration Register | EALLOW | Go |
| 53h | ADCPPB3OFFREF | ADC PPB3 Offset Reference Register | | Go |
| 54h | ADCPPB3TRIPHI | ADC PPB3 Trip High Register | EALLOW | Go |
| 56h | ADCPPB3TRIPLO | ADC PPB3 Trip Low/Trigger Time Stamp Register | EALLOW | Go |
| 58h | ADCPPB4CONFIG | ADC PPB4 Config Register | EALLOW | Go |
| 59h | ADCPPB4STAMP | ADC PPB4 Sample Delay Time Stamp Register | | Go |
| 5Ah | ADCPPB4OFFCAL | ADC PPB4 Offset Calibration Register | EALLOW | Go |
| 5Bh | ADCPPB4OFFREF | ADC PPB4 Offset Reference Register | | Go |
| 5Ch | ADCPPB4TRIPHI | ADC PPB4 Trip High Register | EALLOW | Go |
| 5Eh | ADCPPB4TRIPLO | ADC PPB4 Trip Low/Trigger Time Stamp Register | EALLOW | Go |
| 70h | ADCINLTRIM1 | ADC Linearity Trim 1 Register | EALLOW | Go |
| 72h | ADCINLTRIM2 | ADC Linearity Trim 2 Register | EALLOW | Go |
| 74h | ADCINLTRIM3 | ADC Linearity Trim 3 Register | EALLOW | Go |
| 76h | ADCINLTRIM4 | ADC Linearity Trim 4 Register | EALLOW | Go |
| 78h | ADCINLTRIM5 | ADC Linearity Trim 5 Register | EALLOW | Go |
| 7Ah | ADCINLTRIM6 | ADC Linearity Trim 6 Register | EALLOW | Go |

Complex bit access types are encoded to fit into small table cells. Table 11-38 shows the codes that are used for access types in this section.

**Table 11-38. ADC_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| R-0 | R<br>-0 | Read<br>Returns 0s |
| Write Type | | |
| W | W | Write |
| W1C | W<br>1C | Write<br>1 to clear |
| W1S | W<br>1S | Write<br>1 to set |
| Reset or Default Value | | |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-38. ADC_REGS Access Type Codes (continued)**

| Access Type | Code | Description |
|---|---|---|
| *-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. |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.1 ADCCTL1 Register (Offset = 0h) [Reset = 0000h]

ADCCTL1 is shown in Figure 11-46 and described in Table 11-39.

Return to the Summary Table.

ADC Control 1 Register

**Figure 11-46. 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 11-39. 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.<br>0 ADC is available to sample next channel<br>1 ADC is busy and cannot sample another channel<br>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.<br>When ADCBSY=0: holds the value of the last converted SOC<br>When ADCBSY=1: reflects the SOC currently being processed<br>0h SOC0 is currently processing or was last SOC converted<br>1h SOC1 is currently processing or was last SOC converted<br>2h SOC2 is currently processing or was last SOC converted<br>3h SOC3 is currently processing or was last SOC converted<br>4h SOC4 is currently processing or was last SOC converted<br>5h SOC5 is currently processing or was last SOC converted<br>6h SOC6 is currently processing or was last SOC converted<br>7h SOC7 is currently processing or was last SOC converted<br>8h SOC8 is currently processing or was last SOC converted<br>9h SOC9 is currently processing or was last SOC converted<br>Ah SOC10 is currently processing or was last SOC converted<br>Bh SOC11 is currently processing or was last SOC converted<br>Ch SOC12 is currently processing or was last SOC converted<br>Dh SOC13 is currently processing or was last SOC converted<br>Eh SOC14 is currently processing or was last SOC converted<br>Fh SOC15 is currently processing or was last SOC converted<br>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.<br>0 All analog circuitry inside the core is powered down<br>1 All analog circuitry inside the core is powered up<br>Reset type: SYSRSn |
| 6-3 | RESERVED | R | 0h | Reserved |
| 2 | INTPULSEPOS | R/W | 0h | ADC Interrupt Pulse Position.<br>0 Interrupt pulse generation occurs at the end of the acquisition window<br>1 Interrupt pulse generation occurs at the end of the conversion. Results will latch 1 or more cycles later. Refer to the ADC timing diagrams for exact timings for the specific configurations being used.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-39. ADCCTL1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1-0 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.2 ADCCTL2 Register (Offset = 1h) [Reset = 0000h]

ADCCTL2 is shown in Figure 11-47 and described in Table 11-40.

Return to the Summary Table.

ADC Control 2 Register

**Figure 11-47. ADCCTL2 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | RESERVED | | | |
| R-0h | | | | R-0h | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SIGNALMODE | RESOLUTION | RESERVED | | PRESCALE | | | |
| R/W-0h | R/W-0h | R-0h | | R/W-0h | | | |

**Table 11-40. ADCCTL2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-13 | RESERVED | R | 0h | Reserved |
| 12-8 | RESERVED | R | 0h | Reserved |
| 7 | SIGNALMODE | R/W | 0h | SOC Signaling Mode. Selects the input mode of the converter. Use the AdcSetMode function to change the signal mode.<br>0 Single-ended<br>1 Differential<br>Reset type: SYSRSn |
| 6 | RESOLUTION | R/W | 0h | SOC Conversion Resolution. Selects the resolution of the converter. Use the AdcSetMode function to change the resolution.<br>0 12-bit resolution<br>1 16-bit resolution<br>Reset type: SYSRSn |
| 5-4 | RESERVED | R | 0h | Reserved |
| 3-0 | PRESCALE | R/W | 0h | ADC Clock Prescaler.<br>0000 ADCCLK = Input Clock / 1.0<br>0001 Invalid<br>0010 ADCCLK = Input Clock / 2.0<br>0011 ADCCLK = Input Clock / 2.5<br>0100 ADCCLK = Input Clock / 3.0<br>0101 ADCCLK = Input Clock / 3.5<br>0110 ADCCLK = Input Clock / 4.0<br>0111 ADCCLK = Input Clock / 4.5<br>1000 ADCCLK = Input Clock / 5.0<br>1001 ADCCLK = Input Clock / 5.5<br>1010 ADCCLK = Input Clock / 6.0<br>1011 ADCCLK = Input Clock / 6.5<br>1100 ADCCLK = Input Clock / 7.0<br>1101 ADCCLK = Input Clock / 7.5<br>1110 ADCCLK = Input Clock / 8.0<br>1111 ADCCLK = Input Clock / 8.5<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.3 ADCBURSTCTL Register (Offset = 2h) [Reset = 0000h]

ADCBURSTCTL is shown in Figure 11-48 and described in Table 11-41.

Return to the Summary Table.

ADC Burst Control Register

**Figure 11-48. 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 11-41. 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.<br>0 Burst mode is disabled.<br>1 Burst mode is enabled.<br>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 SOCs 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.<br>0h 1 SOC converted<br>1h 2 SOCs converted<br>2h 3 SOCs converted<br>3h 4 SOCs converted<br>4h 5 SOCs converted<br>5h 6 SOCs converted<br>6h 7 SOCs converted<br>7h 8 SOCs converted<br>8h 9 SOCs converted<br>9h 10 SOCs converted<br>Ah 11 SOCs converted<br>Bh 12 SOCs converted<br>Ch 13 SOCs converted<br>Dh 14 SOCs converted<br>Eh 15 SOCs converted<br>Fh 16 SOCs converted<br>Reset type: SYSRSn |
| 7-6 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-41. 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.<br>Note: SOCFRC1 register can always be used to software trigger SOCs in addition to any hardware trigger configuration.<br>00h BURSTTRIG0 - Software only<br>01h BURSTTRIG1 - CPU1 Timer 0, TINT0n<br>02h BURSTTRIG2 - CPU1 Timer 1, TINT1n<br>03h BURSTTRIG3 - CPU1 Timer 2, TINT2n<br>04h BURSTTRIG4 - GPIO, Input X-Bar INPUT5<br>05h BURSTTRIG5 - ePWM1, ADCSOCA<br>06h BURSTTRIG6 - ePWM1, ADCSOCB<br>07h BURSTTRIG7 - ePWM2, ADCSOCA<br>08h BURSTTRIG8 - ePWM2, ADCSOCB<br>09h BURSTTRIG9 - ePWM3, ADCSOCA<br>0Ah BURSTTRIG10 - ePWM3, ADCSOCB<br>0Bh BURSTTRIG11 - ePWM4, ADCSOCA<br>0Ch BURSTTRIG12 - ePWM4, ADCSOCB<br>0Dh BURSTTRIG13 - ePWM5, ADCSOCA<br>0Eh BURSTTRIG14 - ePWM5, ADCSOCB<br>0Fh BURSTTRIG15 - ePWM6, ADCSOCA<br>10h BURSTTRIG16 - ePWM6, ADCSOCB<br>11h BURSTTRIG17 - ePWM7, ADCSOCA<br>12h BURSTTRIG18 - ePWM7, ADCSOCB<br>13h BURSTTRIG19 - ePWM8, ADCSOCA<br>14h BURSTTRIG20 - ePWM8, ADCSOCB<br>15h BURSTTRIG21 - ePWM9, ADCSOCA<br>16h BURSTTRIG22 - ePWM9, ADCSOCB<br>17h BURSTTRIG23 - ePWM10, ADCSOCA<br>18h BURSTTRIG24 - ePWM10, ADCSOCB<br>19h BURSTTRIG25 - ePWM11, ADCSOCA<br>1Ah BURSTTRIG26 - ePWM11, ADCSOCB<br>1Bh BURSTTRIG27 - ePWM12, ADCSOCA<br>1Ch BURSTTRIG28 - ePWM12, ADCSOCB<br>1Dh BURSTTRIG29 - CPU2 Timer 0, TINT0n<br>1Eh BURSTTRIG30 - CPU2 Timer 1, TINT1n<br>1Fh BURSTTRIG31 - CPU2 Timer 2, TINT2n<br>20h - 3Fh - Reserved<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.4 ADCINTFLG Register (Offset = 3h) [Reset = 0000h]

ADCINTFLG is shown in Figure 11-49 and described in Table 11-42.

Return to the Summary Table.

ADC Interrupt Flag Register

**Figure 11-49. 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 11-42. 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.<br>0 No ADC interrupt pulse generated<br>1 ADC interrupt pulse generated<br>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.<br>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.<br>0 No ADC interrupt pulse generated<br>1 ADC interrupt pulse generated<br>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.<br>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.<br>0 No ADC interrupt pulse generated<br>1 ADC interrupt pulse generated<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-42. 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.<br>0 No ADC interrupt pulse generated<br>1 ADC interrupt pulse generated<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.5 ADCINTFLGCLR Register (Offset = 4h) [Reset = 0000h]

ADCINTFLGCLR is shown in Figure 11-50 and described in Table 11-43.

Return to the Summary Table.

ADC Interrupt Flag Clear Register

**Figure 11-50. 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 11-43. 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.<br>0 No action<br>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<br>Reset type: SYSRSn |
| 2 | ADCINT3 | R-0/W1C | 0h | ADC Interrupt 3 Flag Clear. Reads return 0.<br>0 No action<br>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<br>Reset type: SYSRSn |
| 1 | ADCINT2 | R-0/W1C | 0h | ADC Interrupt 2 Flag Clear. Reads return 0.<br>0 No action<br>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<br>Reset type: SYSRSn |
| 0 | ADCINT1 | R-0/W1C | 0h | ADC Interrupt 1 Flag Clear. Reads return 0.<br>0 No action<br>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<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.6 ADCINTOVF Register (Offset = 5h) [Reset = 0000h]

ADCINTOVF is shown in Figure 11-51 and described in Table 11-44.

Return to the Summary Table.

ADC Interrupt Overflow Register

**Figure 11-51. 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 11-44. ADCINTOVF Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-4 | RESERVED | R | 0h | Reserved |
| 3 | ADCINT4 | R | 0h | ADC Interrupt 4 Overflow Flags<br>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.<br>0 No ADC interrupt overflow event detected.<br>1 ADC Interrupt overflow event detected.<br>The overflow bit does not care about the continuous mode bit state. An overflow condition is generated irrespective of this mode selection.<br>Reset type: SYSRSn |
| 2 | ADCINT3 | R | 0h | ADC Interrupt 3 Overflow Flags<br>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.<br>0 No ADC interrupt overflow event detected.<br>1 ADC Interrupt overflow event detected.<br>The overflow bit does not care about the continuous mode bit state. An overflow condition is generated irrespective of this mode selection.<br>Reset type: SYSRSn |
| 1 | ADCINT2 | R | 0h | ADC Interrupt 2 Overflow Flags<br>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.<br>0 No ADC interrupt overflow event detected.<br>1 ADC Interrupt overflow event detected.<br>The overflow bit does not care about the continuous mode bit state. An overflow condition is generated irrespective of this mode selection.<br>Reset type: SYSRSn |
| 0 | ADCINT1 | R | 0h | ADC Interrupt 1 Overflow Flags<br>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.<br>0 No ADC interrupt overflow event detected.<br>1 ADC Interrupt overflow event detected.<br>The overflow bit does not care about the continuous mode bit state. An overflow condition is generated irrespective of this mode selection.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.7 ADCINTOVFCLR Register (Offset = 6h) [Reset = 0000h]

ADCINTOVFCLR is shown in Figure 11-52 and described in Table 11-45.

Return to the Summary Table.

ADC Interrupt Overflow Clear Register

**Figure 11-52. 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 11-45. 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<br>0 No action.<br>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.<br>Reset type: SYSRSn |
| 2 | ADCINT3 | R-0/W1C | 0h | ADC Interrupt 3 Overflow Clear Bits<br>0 No action.<br>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.<br>Reset type: SYSRSn |
| 1 | ADCINT2 | R-0/W1C | 0h | ADC Interrupt 2 Overflow Clear Bits<br>0 No action.<br>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.<br>Reset type: SYSRSn |
| 0 | ADCINT1 | R-0/W1C | 0h | ADC Interrupt 1 Overflow Clear Bits<br>0 No action.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.8 ADCINTSEL1N2 Register (Offset = 7h) [Reset = 0000h]

ADCINTSEL1N2 is shown in Figure 11-53 and described in Table 11-46.

Return to the Summary Table.

ADC Interrupt 1 and 2 Selection Register

**Figure 11-53. 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 11-46. ADCINTSEL1N2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RESERVED | R | 0h | Reserved |
| 14 | INT2CONT | R/W | 0h | ADCINT2 Continue to Interrupt Mode<br>0 No further ADCINT2 pulses are generated until ADCINT2 flag (in ADCINTFLG register) is cleared by user.<br>1 ADCINT2 pulses are generated whenever an EOC pulse is generated irrespective of whether the flag bit is cleared or not.<br>Reset type: SYSRSn |
| 13 | INT2E | R/W | 0h | ADCINT2 Interrupt Enable<br>0 ADCINT2 is disabled<br>1 ADCINT2 is enabled<br>Reset type: SYSRSn |
| 12 | RESERVED | R | 0h | Reserved |
| 11-8 | INT2SEL | R/W | 0h | ADCINT2 EOC Source Select<br>0h EOC0 is trigger for ADCINT2<br>1h EOC1 is trigger for ADCINT2<br>2h EOC2 is trigger for ADCINT2<br>3h EOC3 is trigger for ADCINT2<br>4h EOC4 is trigger for ADCINT2<br>5h EOC5 is trigger for ADCINT2<br>6h EOC6 is trigger for ADCINT2<br>7h EOC7 is trigger for ADCINT2<br>8h EOC8 is trigger for ADCINT2<br>9h EOC9 is trigger for ADCINT2<br>Ah EOC10 is trigger for ADCINT2<br>Bh EOC11 is trigger for ADCINT2<br>Ch EOC12 is trigger for ADCINT2<br>Dh EOC13 is trigger for ADCINT2<br>Eh EOC14 is trigger for ADCINT2<br>Fh EOC15 is trigger for ADCINT2<br>Reset type: SYSRSn |
| 7 | RESERVED | R | 0h | Reserved |
| 6 | INT1CONT | R/W | 0h | ADCINT1 Continue to Interrupt Mode<br>0 No further ADCINT1 pulses are generated until ADCINT1 flag (in ADCINTFLG register) is cleared by user.<br>1 ADCINT1 pulses are generated whenever an EOC pulse is generated irrespective of whether the flag bit is cleared or not.<br>Reset type: SYSRSn |
| 5 | INT1E | R/W | 0h | ADCINT1 Interrupt Enable<br>0 ADCINT1 is disabled<br>1 ADCINT1 is enabled<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-46. 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<br>0h EOC0 is trigger for ADCINT1<br>1h EOC1 is trigger for ADCINT1<br>2h EOC2 is trigger for ADCINT1<br>3h EOC3 is trigger for ADCINT1<br>4h EOC4 is trigger for ADCINT1<br>5h EOC5 is trigger for ADCINT1<br>6h EOC6 is trigger for ADCINT1<br>7h EOC7 is trigger for ADCINT1<br>8h EOC8 is trigger for ADCINT1<br>9h EOC9 is trigger for ADCINT1<br>Ah EOC10 is trigger for ADCINT1<br>Bh EOC11 is trigger for ADCINT1<br>Ch EOC12 is trigger for ADCINT1<br>Dh EOC13 is trigger for ADCINT1<br>Eh EOC14 is trigger for ADCINT1<br>Fh EOC15 is trigger for ADCINT1<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.9 ADCINTSEL3N4 Register (Offset = 8h) [Reset = 0000h]

ADCINTSEL3N4 is shown in Figure 11-54 and described in Table 11-47.

Return to the Summary Table.

ADC Interrupt 3 and 4 Selection Register

**Figure 11-54. 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 11-47. ADCINTSEL3N4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RESERVED | R | 0h | Reserved |
| 14 | INT4CONT | R/W | 0h | ADCINT4 Continue to Interrupt Mode<br>0 No further ADCINT4 pulses are generated until ADCINT4 flag (in ADCINTFLG register) is cleared by user.<br>1 ADCINT4 pulses are generated whenever an EOC pulse is generated irrespective of whether the flag bit is cleared or not.<br>Reset type: SYSRSn |
| 13 | INT4E | R/W | 0h | ADCINT4 Interrupt Enable<br>0 ADCINT4 is disabled<br>1 ADCINT4 is enabled<br>Reset type: SYSRSn |
| 12 | RESERVED | R | 0h | Reserved |
| 11-8 | INT4SEL | R/W | 0h | ADCINT4 EOC Source Select<br>0h EOC0 is trigger for ADCINT4<br>1h EOC1 is trigger for ADCINT4<br>2h EOC2 is trigger for ADCINT4<br>3h EOC3 is trigger for ADCINT4<br>4h EOC4 is trigger for ADCINT4<br>5h EOC5 is trigger for ADCINT4<br>6h EOC6 is trigger for ADCINT4<br>7h EOC7 is trigger for ADCINT4<br>8h EOC8 is trigger for ADCINT4<br>9h EOC9 is trigger for ADCINT4<br>Ah EOC10 is trigger for ADCINT4<br>Bh EOC11 is trigger for ADCINT4<br>Ch EOC12 is trigger for ADCINT4<br>Dh EOC13 is trigger for ADCINT4<br>Eh EOC14 is trigger for ADCINT4<br>Fh EOC15 is trigger for ADCINT4<br>Reset type: SYSRSn |
| 7 | RESERVED | R | 0h | Reserved |
| 6 | INT3CONT | R/W | 0h | ADCINT3 Continue to Interrupt Mode<br>0 No further ADCINT3 pulses are generated until ADCINT3 flag (in ADCINTFLG register) is cleared by user.<br>1 ADCINT3 pulses are generated whenever an EOC pulse is generated irrespective of whether the flag bit is cleared or not.<br>Reset type: SYSRSn |
| 5 | INT3E | R/W | 0h | ADCINT3 Interrupt Enable<br>0 ADCINT3 is disabled<br>1 ADCINT3 is enabled<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-47. 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<br>0h EOC0 is trigger for ADCINT3<br>1h EOC1 is trigger for ADCINT3<br>2h EOC2 is trigger for ADCINT3<br>3h EOC3 is trigger for ADCINT3<br>4h EOC4 is trigger for ADCINT3<br>5h EOC5 is trigger for ADCINT3<br>6h EOC6 is trigger for ADCINT3<br>7h EOC7 is trigger for ADCINT3<br>8h EOC8 is trigger for ADCINT3<br>9h EOC9 is trigger for ADCINT3<br>Ah EOC10 is trigger for ADCINT3<br>Bh EOC11 is trigger for ADCINT3<br>Ch EOC12 is trigger for ADCINT3<br>Dh EOC13 is trigger for ADCINT3<br>Eh EOC14 is trigger for ADCINT3<br>Fh EOC15 is trigger for ADCINT3<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.10 ADCSOCPRICTL Register (Offset = 9h) [Reset = 0200h]

ADCSOCPRICTL is shown in Figure 11-55 and described in Table 11-48.

Return to the Summary Table.

ADC SOC Priority Control Register

**Figure 11-55. 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 11-48. 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.<br>00h SOC0 was last round robin SOC to convert, SOC1 is highest round robin priority.<br>01h SOC1 was last round robin SOC to convert, SOC2 is highest round robin priority.<br>02h SOC2 was last round robin SOC to convert, SOC3 is highest round robin priority.<br>03h SOC3 was last round robin SOC to convert, SOC4 is highest round robin priority.<br>04h SOC4 was last round robin SOC to convert, SOC5 is highest round robin priority.<br>05h SOC5 was last round robin SOC to convert, SOC6 is highest round robin priority.<br>06h SOC6 was last round robin SOC to convert, SOC7 is highest round robin priority.<br>07h SOC7 was last round robin SOC to convert, SOC8 is highest round robin priority.<br>08h SOC8 was last round robin SOC to convert, SOC9 is highest round robin priority.<br>09h SOC9 was last round robin SOC to convert, SOC10 is highest round robin priority.<br>0Ah SOC10 was last round robin SOC to convert, SOC11 is highest round robin priority.<br>0Bh SOC11 was last round robin SOC to convert, SOC12 is highest round robin priority.<br>0Ch SOC12 was last round robin SOC to convert, SOC13 is highest round robin priority.<br>0Dh SOC13 was last round robin SOC to convert, SOC14 is highest round robin priority.<br>0Eh SOC14 was last round robin SOC to convert, SOC15 is highest round robin priority.<br>0Fh SOC15 was last round robin SOC to convert, SOC0 is highest round robin priority.<br>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.<br>Others Invalid value.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-48. ADCSOCPRICTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 4-0 | SOCPRIORITY | R/W | 0h | SOC Priority<br>Determines the cutoff point for priority mode and round robin arbitration for SOCx<br>00h SOC priority is handled in round robin mode for all channels.<br>01h SOC0 is high priority, rest of channels are in round robin mode.<br>02h SOC0-SOC1 are high priority, SOC2-SOC15 are in round robin mode.<br>03h SOC0-SOC2 are high priority, SOC3-SOC15 are in round robin mode.<br>04h SOC0-SOC3 are high priority, SOC4-SOC15 are in round robin mode.<br>05h SOC0-SOC4 are high priority, SOC5-SOC15 are in round robin mode.<br>06h SOC0-SOC5 are high priority, SOC6-SOC15 are in round robin mode.<br>07h SOC0-SOC6 are high priority, SOC7-SOC15 are in round robin mode.<br>08h SOC0-SOC7 are high priority, SOC8-SOC15 are in round robin mode.<br>09h SOC0-SOC8 are high priority, SOC9-SOC15 are in round robin mode.<br>0Ah SOC0-SOC9 are high priority, SOC10-SOC15 are in round robin mode.<br>0Bh SOC0-SOC10 are high priority, SOC11-SOC15 are in round robin mode.<br>0Ch SOC0-SOC11 are high priority, SOC12-SOC15 are in round robin mode.<br>0Dh SOC0-SOC12 are high priority, SOC13-SOC15 are in round robin mode.<br>0Eh SOC0-SOC13 are high priority, SOC14-SOC15 are in round robin mode.<br>0Fh SOC0-SOC14 are high priority, SOC15 is in round robin mode.<br>10h All SOCs are in high priority mode, arbitrated by SOC number.<br>Others Invalid selection.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.11 ADCINTSOCSEL1 Register (Offset = Ah) [Reset = 0000h]

ADCINTSOCSEL1 is shown in Figure 11-56 and described in Table 11-49.

Return to the Summary Table.

ADC Interrupt SOC Selection 1 Register

**Figure 11-56. 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 11-49. 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.<br>00 No ADCINT will trigger SOC7. TRIGSEL field alone determines SOC0 trigger.<br>01 ADCINT1 will trigger SOC7.<br>10 ADCINT2 will trigger SOC7.<br>11 Invalid selection.<br>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.<br>00 No ADCINT will trigger SOC6. TRIGSEL field alone determines SOC0 trigger.<br>01 ADCINT1 will trigger SOC6.<br>10 ADCINT2 will trigger SOC6.<br>11 Invalid selection.<br>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.<br>00 No ADCINT will trigger SOC5. TRIGSEL field alone determines SOC0 trigger.<br>01 ADCINT1 will trigger SOC5.<br>10 ADCINT2 will trigger SOC5.<br>11 Invalid selection.<br>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.<br>00 No ADCINT will trigger SOC4. TRIGSEL field alone determines SOC0 trigger.<br>01 ADCINT1 will trigger SOC4.<br>10 ADCINT2 will trigger SOC4.<br>11 Invalid selection.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-49. 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.<br>00 No ADCINT will trigger SOC3. TRIGSEL field alone determines SOC0 trigger.<br>01 ADCINT1 will trigger SOC3.<br>10 ADCINT2 will trigger SOC3.<br>11 Invalid selection.<br>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.<br>00 No ADCINT will trigger SOC2. TRIGSEL field alone determines SOC0 trigger.<br>01 ADCINT1 will trigger SOC2.<br>10 ADCINT2 will trigger SOC2.<br>11 Invalid selection.<br>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.<br>00 No ADCINT will trigger SOC1. TRIGSEL field alone determines SOC0 trigger.<br>01 ADCINT1 will trigger SOC1.<br>10 ADCINT2 will trigger SOC1.<br>11 Invalid selection.<br>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.<br>00 No ADCINT will trigger SOC0. TRIGSEL field alone determines SOC0 trigger.<br>01 ADCINT1 will trigger SOC0.<br>10 ADCINT2 will trigger SOC0.<br>11 Invalid selection.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.12 ADCINTSOCSEL2 Register (Offset = Bh) [Reset = 0000h]

ADCINTSOCSEL2 is shown in Figure 11-57 and described in Table 11-50.

Return to the Summary Table.

ADC Interrupt SOC Selection 2 Register

**Figure 11-57. 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 11-50. 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.<br>00 No ADCINT will trigger SOC15. TRIGSEL field alone determines SOC0 trigger.<br>01 ADCINT1 will trigger SOC15.<br>10 ADCINT2 will trigger SOC15.<br>11 Invalid selection.<br>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.<br>00 No ADCINT will trigger SOC14. TRIGSEL field alone determines SOC0 trigger.<br>01 ADCINT1 will trigger SOC14.<br>10 ADCINT2 will trigger SOC14.<br>11 Invalid selection.<br>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.<br>00 No ADCINT will trigger SOC13. TRIGSEL field alone determines SOC0 trigger.<br>01 ADCINT1 will trigger SOC13.<br>10 ADCINT2 will trigger SOC13.<br>11 Invalid selection.<br>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.<br>00 No ADCINT will trigger SOC12. TRIGSEL field alone determines SOC0 trigger.<br>01 ADCINT1 will trigger SOC12.<br>10 ADCINT2 will trigger SOC12.<br>11 Invalid selection.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-50. 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.<br>00 No ADCINT will trigger SOC11. TRIGSEL field alone determines SOC0 trigger.<br>01 ADCINT1 will trigger SOC11.<br>10 ADCINT2 will trigger SOC11.<br>11 Invalid selection.<br>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.<br>00 No ADCINT will trigger SOC10. TRIGSEL field alone determines SOC0 trigger.<br>01 ADCINT1 will trigger SOC10.<br>10 ADCINT2 will trigger SOC10.<br>11 Invalid selection.<br>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.<br>00 No ADCINT will trigger SOC9. TRIGSEL field alone determines SOC0 trigger.<br>01 ADCINT1 will trigger SOC9.<br>10 ADCINT2 will trigger SOC9.<br>11 Invalid selection.<br>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.<br>00 No ADCINT will trigger SOC8. TRIGSEL field alone determines SOC0 trigger.<br>01 ADCINT1 will trigger SOC8.<br>10 ADCINT2 will trigger SOC8.<br>11 Invalid selection.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.13 ADCSOCFLG1 Register (Offset = Ch) [Reset = 0000h]

ADCSOCFLG1 is shown in Figure 11-58 and described in Table 11-51.

Return to the Summary Table.

ADC SOC Flag 1 Register

**Figure 11-58. 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 11-51. ADCSOCFLG1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | SOC15 | R | 0h | SOC15 Start of Conversion Flag. Indicates the state of SOC15 conversions.<br>0 No sample pending for SOC15.<br>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.<br>Reset type: SYSRSn |
| 14 | SOC14 | R | 0h | SOC14 Start of Conversion Flag. Indicates the state of SOC14 conversions.<br>0 No sample pending for SOC14.<br>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.<br>Reset type: SYSRSn |
| 13 | SOC13 | R | 0h | SOC13 Start of Conversion Flag. Indicates the state of SOC13 conversions.<br>0 No sample pending for SOC13.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-51. ADCSOCFLG1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 12 | SOC12 | R | 0h | SOC12 Start of Conversion Flag. Indicates the state of SOC12 conversions.<br>0 No sample pending for SOC12.<br>1 Trigger has been received and sample is pending for SOC12. This bit will be automatically cleared when the SOC12 conversion is started. If contention exists where this bit receives both a request to set and a request to clear on the same cycle, regardless of the source of either, this bit will be set and the request to clear will be ignored. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether this bit was previously set or not.<br>Reset type: SYSRSn |
| 11 | SOC11 | R | 0h | SOC11 Start of Conversion Flag. Indicates the state of SOC11 conversions.<br>0 No sample pending for SOC11.<br>1 Trigger has been received and sample is pending for SOC11. 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.<br>Reset type: SYSRSn |
| 10 | SOC10 | R | 0h | SOC10 Start of Conversion Flag. Indicates the state of SOC10 conversions.<br>0 No sample pending for SOC10.<br>1 Trigger has been received and sample is pending for SOC10. 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.<br>Reset type: SYSRSn |
| 9 | SOC9 | R | 0h | SOC9 Start of Conversion Flag. Indicates the state of SOC9 conversions.<br>0 No sample pending for SOC9.<br>1 Trigger has been received and sample is pending for SOC9. 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.<br>Reset type: SYSRSn |
| 8 | SOC8 | R | 0h | SOC8 Start of Conversion Flag. Indicates the state of SOC8 conversions.<br>0 No sample pending for SOC8.<br>1 Trigger has been received and sample is pending for SOC8. 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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-51. ADCSOCFLG1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 7 | SOC7 | R | 0h | SOC7 Start of Conversion Flag. Indicates the state of SOC7 conversions.<br>0 No sample pending for SOC7.<br>1 Trigger has been received and sample is pending for SOC7.<br>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.<br>Reset type: SYSRSn |
| 6 | SOC6 | R | 0h | SOC6 Start of Conversion Flag. Indicates the state of SOC6 conversions.<br>0 No sample pending for SOC6.<br>1 Trigger has been received and sample is pending for SOC6.<br>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.<br>Reset type: SYSRSn |
| 5 | SOC5 | R | 0h | SOC5 Start of Conversion Flag. Indicates the state of SOC5 conversions.<br>0 No sample pending for SOC5.<br>1 Trigger has been received and sample is pending for SOC5.<br>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.<br>Reset type: SYSRSn |
| 4 | SOC4 | R | 0h | SOC4 Start of Conversion Flag. Indicates the state of SOC4 conversions.<br>0 No sample pending for SOC4.<br>1 Trigger has been received and sample is pending for SOC4.<br>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.<br>Reset type: SYSRSn |
| 3 | SOC3 | R | 0h | SOC3 Start of Conversion Flag. Indicates the state of SOC3 conversions.<br>0 No sample pending for SOC3.<br>1 Trigger has been received and sample is pending for SOC3.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-51. ADCSOCFLG1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 2 | SOC2 | R | 0h | SOC2 Start of Conversion Flag. Indicates the state of SOC2 conversions.<br>0 No sample pending for SOC2.<br>1 Trigger has been received and sample is pending for SOC2.<br>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.<br>Reset type: SYSRSn |
| 1 | SOC1 | R | 0h | SOC1 Start of Conversion Flag. Indicates the state of SOC1 conversions.<br>0 No sample pending for SOC1.<br>1 Trigger has been received and sample is pending for SOC1.<br>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.<br>Reset type: SYSRSn |
| 0 | SOC0 | R | 0h | SOC0 Start of Conversion Flag. Indicates the state of SOC0 conversions.<br>0 No sample pending for SOC0.<br>1 Trigger has been received and sample is pending for SOC0.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.14 ADCSOCFRC1 Register (Offset = Dh) [Reset = 0000h]

ADCSOCFRC1 is shown in Figure 11-59 and described in Table 11-52.

Return to the Summary Table.

ADC SOC Force 1 Register

**Figure 11-59. 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 11-52. ADCSOCFRC1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | SOC15 | R-0/W1S | 0h | 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.<br>0 No action.<br>1 Force SOC15 flag bit to 1. This will cause a conversion to start once priority is given to SOC15.<br>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.<br>Reset type: SYSRSn |
| 14 | SOC14 | R-0/W1S | 0h | 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.<br>0 No action.<br>1 Force SOC14 flag bit to 1. This will cause a conversion to start once priority is given to SOC14.<br>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.<br>Reset type: SYSRSn |
| 13 | SOC13 | R-0/W1S | 0h | 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.<br>0 No action.<br>1 Force SOC13 flag bit to 1. This will cause a conversion to start once priority is given to SOC13.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-52. ADCSOCFRC1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 12 | SOC12 | R-0/W1S | 0h | 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.<br>0 No action.<br>1 Force SOC12 flag bit to 1. This will cause a conversion to start once priority is given to SOC12.<br>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.<br>Reset type: SYSRSn |
| 11 | SOC11 | R-0/W1S | 0h | 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.<br>0 No action.<br>1 Force SOC11 flag bit to 1. This will cause a conversion to start once priority is given to SOC11.<br>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.<br>Reset type: SYSRSn |
| 10 | SOC10 | R-0/W1S | 0h | 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.<br>0 No action.<br>1 Force SOC10 flag bit to 1. This will cause a conversion to start once priority is given to SOC10.<br>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.<br>Reset type: SYSRSn |
| 9 | SOC9 | R-0/W1S | 0h | 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.<br>0 No action.<br>1 Force SOC9 flag bit to 1. This will cause a conversion to start once priority is given to SOC9.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-52. ADCSOCFRC1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 8 | SOC8 | R-0/W1S | 0h | 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.<br>0 No action.<br>1 Force SOC8 flag bit to 1. This will cause a conversion to start once priority is given to SOC8.<br>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.<br>Reset type: SYSRSn |
| 7 | SOC7 | R-0/W1S | 0h | 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.<br>0 No action.<br>1 Force SOC7 flag bit to 1. This will cause a conversion to start once priority is given to SOC7.<br>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.<br>Reset type: SYSRSn |
| 6 | SOC6 | R-0/W1S | 0h | 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.<br>0 No action.<br>1 Force SOC6 flag bit to 1. This will cause a conversion to start once priority is given to SOC6.<br>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.<br>Reset type: SYSRSn |
| 5 | SOC5 | R-0/W1S | 0h | 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.<br>0 No action.<br>1 Force SOC5 flag bit to 1. This will cause a conversion to start once priority is given to SOC5.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-52. ADCSOCFRC1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 4 | SOC4 | R-0/W1S | 0h | 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.<br>0 No action.<br>1 Force SOC4 flag bit to 1. This will cause a conversion to start once priority is given to SOC4.<br>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.<br>Reset type: SYSRSn |
| 3 | SOC3 | R-0/W1S | 0h | 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.<br>0 No action.<br>1 Force SOC3 flag bit to 1. This will cause a conversion to start once priority is given to SOC3.<br>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.<br>Reset type: SYSRSn |
| 2 | SOC2 | R-0/W1S | 0h | 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.<br>0 No action.<br>1 Force SOC2 flag bit to 1. This will cause a conversion to start once priority is given to SOC2.<br>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.<br>Reset type: SYSRSn |
| 1 | SOC1 | R-0/W1S | 0h | 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.<br>0 No action.<br>1 Force SOC1 flag bit to 1. This will cause a conversion to start once priority is given to SOC1.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-52. ADCSOCFRC1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | SOC0 | R-0/W1S | 0h | 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.<br>0 No action.<br>1 Force SOC0 flag bit to 1. This will cause a conversion to start once priority is given to SOC0.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.15 ADCSOCOVF1 Register (Offset = Eh) [Reset = 0000h]

ADCSOCOVF1 is shown in Figure 11-60 and described in Table 11-53.

Return to the Summary Table.

ADC SOC Overflow 1 Register

**Figure 11-60. 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 11-53. 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.<br>0 No SOC15 event overflow.<br>1 SOC15 event overflow.<br>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.<br>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.<br>0 No SOC14 event overflow.<br>1 SOC14 event overflow.<br>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.<br>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.<br>0 No SOC13 event overflow.<br>1 SOC13 event overflow.<br>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.<br>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.<br>0 No SOC12 event overflow.<br>1 SOC12 event overflow.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-53. ADCSOCOVF1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 11 | SOC11 | R | 0h | SOC11 Start of Conversion Overflow Flag. Indicates an SOC11 event was generated in hardware while an existing SOC11 event was already pending.<br>0 No SOC11 event overflow.<br>1 SOC11 event overflow.<br>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.<br>Reset type: SYSRSn |
| 10 | SOC10 | R | 0h | SOC10 Start of Conversion Overflow Flag. Indicates an SOC10 event was generated in hardware while an existing SOC10 event was already pending.<br>0 No SOC10 event overflow.<br>1 SOC10 event overflow.<br>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.<br>Reset type: SYSRSn |
| 9 | SOC9 | R | 0h | SOC9 Start of Conversion Overflow Flag. Indicates an SOC9 event was generated in hardware while an existing SOC9 event was already pending.<br>0 No SOC9 event overflow.<br>1 SOC9 event overflow.<br>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.<br>Reset type: SYSRSn |
| 8 | SOC8 | R | 0h | SOC8 Start of Conversion Overflow Flag. Indicates an SOC8 event was generated in hardware while an existing SOC8 event was already pending.<br>0 No SOC8 event overflow.<br>1 SOC8 event overflow.<br>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.<br>Reset type: SYSRSn |
| 7 | SOC7 | R | 0h | SOC7 Start of Conversion Overflow Flag. Indicates an SOC7 event was generated in hardware while an existing SOC7 event was already pending.<br>0 No SOC7 event overflow.<br>1 SOC7 event overflow.<br>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.<br>Reset type: SYSRSn |
| 6 | SOC6 | R | 0h | SOC6 Start of Conversion Overflow Flag. Indicates an SOC6 event was generated in hardware while an existing SOC6 event was already pending.<br>0 No SOC6 event overflow.<br>1 SOC6 event overflow.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-53. ADCSOCOVF1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 5 | SOC5 | R | 0h | SOC5 Start of Conversion Overflow Flag. Indicates an SOC5 event was generated in hardware while an existing SOC5 event was already pending.<br>0 No SOC5 event overflow.<br>1 SOC5 event overflow.<br>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.<br>Reset type: SYSRSn |
| 4 | SOC4 | R | 0h | SOC4 Start of Conversion Overflow Flag. Indicates an SOC4 event was generated in hardware while an existing SOC4 event was already pending.<br>0 No SOC4 event overflow.<br>1 SOC4 event overflow.<br>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.<br>Reset type: SYSRSn |
| 3 | SOC3 | R | 0h | SOC3 Start of Conversion Overflow Flag. Indicates an SOC3 event was generated in hardware while an existing SOC3 event was already pending.<br>0 No SOC3 event overflow.<br>1 SOC3 event overflow.<br>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.<br>Reset type: SYSRSn |
| 2 | SOC2 | R | 0h | SOC2 Start of Conversion Overflow Flag. Indicates an SOC2 event was generated in hardware while an existing SOC2 event was already pending.<br>0 No SOC2 event overflow.<br>1 SOC2 event overflow.<br>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.<br>Reset type: SYSRSn |
| 1 | SOC1 | R | 0h | SOC1 Start of Conversion Overflow Flag. Indicates an SOC1 event was generated in hardware while an existing SOC1 event was already pending.<br>0 No SOC1 event overflow.<br>1 SOC1 event overflow.<br>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.<br>Reset type: SYSRSn |
| 0 | SOC0 | R | 0h | SOC0 Start of Conversion Overflow Flag. Indicates an SOC0 event was generated in hardware while an existing SOC0 event was already pending.<br>0 No SOC0 event overflow.<br>1 SOC0 event overflow.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.16 ADCSOCOVFCLR1 Register (Offset = Fh) [Reset = 0000h]

ADCSOCOVFCLR1 is shown in Figure 11-61 and described in Table 11-54.

Return to the Summary Table.

ADC SOC Overflow Clear 1 Register

**Figure 11-61. 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 11-54. 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.<br>0 No action.<br>1 Clear SOC15 overflow flag.<br>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..<br>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.<br>0 No action.<br>1 Clear SOC14 overflow flag.<br>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..<br>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.<br>0 No action.<br>1 Clear SOC13 overflow flag.<br>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..<br>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.<br>0 No action.<br>1 Clear SOC12 overflow flag.<br>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..<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-54. ADCSOCOVFCLR1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 11 | SOC11 | R-0/W1S | 0h | SOC11 Clear Start of Conversion Overflow Bit. Writing a 1 will clear the SOC11 overflow flag in the ADCSOCOVF1 register. Writes of 0 are ignored. Reads will always return a 0.<br>0 No action.<br>1 Clear SOC11 overflow flag.<br>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..<br>Reset type: SYSRSn |
| 10 | SOC10 | R-0/W1S | 0h | 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.<br>0 No action.<br>1 Clear SOC10 overflow flag.<br>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..<br>Reset type: SYSRSn |
| 9 | SOC9 | R-0/W1S | 0h | 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.<br>0 No action.<br>1 Clear SOC9 overflow flag.<br>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..<br>Reset type: SYSRSn |
| 8 | SOC8 | R-0/W1S | 0h | 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.<br>0 No action.<br>1 Clear SOC8 overflow flag.<br>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..<br>Reset type: SYSRSn |
| 7 | SOC7 | R-0/W1S | 0h | 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.<br>0 No action.<br>1 Clear SOC7 overflow flag.<br>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..<br>Reset type: SYSRSn |
| 6 | SOC6 | R-0/W1S | 0h | 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.<br>0 No action.<br>1 Clear SOC6 overflow flag.<br>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..<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-54. ADCSOCOVFCLR1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 5 | SOC5 | R-0/W1S | 0h | 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.<br>0 No action.<br>1 Clear SOC5 overflow flag.<br>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..<br>Reset type: SYSRSn |
| 4 | SOC4 | R-0/W1S | 0h | 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.<br>0 No action.<br>1 Clear SOC4 overflow flag.<br>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..<br>Reset type: SYSRSn |
| 3 | SOC3 | R-0/W1S | 0h | 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.<br>0 No action.<br>1 Clear SOC3 overflow flag.<br>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..<br>Reset type: SYSRSn |
| 2 | SOC2 | R-0/W1S | 0h | 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.<br>0 No action.<br>1 Clear SOC2 overflow flag.<br>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..<br>Reset type: SYSRSn |
| 1 | SOC1 | R-0/W1S | 0h | 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.<br>0 No action.<br>1 Clear SOC1 overflow flag.<br>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..<br>Reset type: SYSRSn |
| 0 | SOC0 | R-0/W1S | 0h | 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.<br>0 No action.<br>1 Clear SOC0 overflow flag.<br>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..<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.17 ADCSOC0CTL Register (Offset = 10h) [Reset = 00000000h]

ADCSOC0CTL is shown in Figure 11-62 and described in Table 11-55.

Return to the Summary Table.

ADC SOC0 Control Register

**Figure 11-62. ADCSOC0CTL Register**

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

**Table 11-55. ADCSOC0CTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-25 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-55. ADCSOC0CTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 24-20 | TRIGSEL | R/W | 0h | SOC0 Trigger Source Select. Along with the SOC0 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC0 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.<br>Note: SOCFRC1 register can always be used to software trigger SOCs in addition to any hardware trigger configuration.<br>00h ADCTRIG0 - Software only<br>01h ADCTRIG1 - CPU1 Timer 0, TINT0n<br>02h ADCTRIG2 - CPU1 Timer 1, TINT1n<br>03h ADCTRIG3 - CPU1 Timer 2, TINT2n<br>04h ADCTRIG4 - GPIO, ADCEXTSOC<br>05h ADCTRIG5 - ePWM1, ADCSOCA<br>06h ADCTRIG6 - ePWM1, ADCSOCB<br>07h ADCTRIG7 - ePWM2, ADCSOCA<br>08h ADCTRIG8 - ePWM2, ADCSOCB<br>09h ADCTRIG9 - ePWM3, ADCSOCA<br>0Ah ADCTRIG10 - ePWM3, ADCSOCB<br>0Bh ADCTRIG11 - ePWM4, ADCSOCA<br>0Ch ADCTRIG12 - ePWM4, ADCSOCB<br>0Dh ADCTRIG13 - ePWM5, ADCSOCA<br>0Eh ADCTRIG14 - ePWM5, ADCSOCB<br>0Fh ADCTRIG15 - ePWM6, ADCSOCA<br>10h ADCTRIG16 - ePWM6, ADCSOCB<br>11h ADCTRIG17 - ePWM7, ADCSOCA<br>12h ADCTRIG18 - ePWM7, ADCSOCB<br>13h ADCTRIG19 - ePWM8, ADCSOCA<br>14h ADCTRIG20 - ePWM8, ADCSOCB<br>15h ADCTRIG21 - ePWM9, ADCSOCA<br>16h ADCTRIG22 - ePWM9, ADCSOCB<br>17h ADCTRIG23 - ePWM10, ADCSOCA<br>18h ADCTRIG24 - ePWM10, ADCSOCB<br>19h ADCTRIG25 - ePWM11, ADCSOCA<br>1Ah ADCTRIG26 - ePWM11, ADCSOCB<br>1Bh ADCTRIG27 - ePWM12, ADCSOCA<br>1Ch ADCTRIG28 - ePWM12, ADCSOCB<br>1Dh ADCTRIG29 - CPU2 Timer 0, TINT0n<br>1Eh ADCTRIG30 - CPU2 Timer 1, TINT1n<br>1Fh ADCTRIG31 - CPU2 Timer 2, TINT2n<br>Reset type: SYSRSn |
| 19 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-55. ADCSOC0CTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 18-15 | CHSEL | R/W | 0h | SOC0 Channel Select. Selects the channel to be converted when SOC0 is received by the ADC.<br>Single-ended Signaling Mode (SIGNALMODE = 0):<br>0h ADCIN0<br>1h ADCIN1<br>2h ADCIN2<br>3h ADCIN3<br>4h ADCIN4<br>5h ADCIN5<br>6h ADCIN6<br>7h ADCIN7<br>8h ADCIN8<br>9h ADCIN9<br>Ah ADCIN10<br>Bh ADCIN11<br>Ch ADCIN12<br>Dh ADCIN13<br>Eh ADCIN14<br>Fh ADCIN15<br>Differential Signaling Mode (SIGNALMODE = 1):<br>0h ADCIN0 (non-inverting) and ADCIN1 (inverting)<br>1h ADCIN0 (non-inverting) and ADCIN1 (inverting)<br>2h ADCIN2 (non-inverting) and ADCIN3 (inverting)<br>3h ADCIN2 (non-inverting) and ADCIN3 (inverting)<br>4h ADCIN4 (non-inverting) and ADCIN5 (inverting)<br>5h ADCIN4 (non-inverting) and ADCIN5 (inverting)<br>6h ADCIN6 (non-inverting) and ADCIN7 (inverting)<br>7h ADCIN6 (non-inverting) and ADCIN7 (inverting)<br>8h ADCIN8 (non-inverting) and ADCIN9 (inverting)<br>9h ADCIN8 (non-inverting) and ADCIN9 (inverting)<br>Ah ADCIN10 (non-inverting) and ADCIN11 (inverting)<br>Bh ADCIN10 (non-inverting) and ADCIN11 (inverting)<br>Ch ADCIN12 (non-inverting) and ADCIN13 (inverting)<br>Dh ADCIN12 (non-inverting) and ADCIN13 (inverting)<br>Eh ADCIN14 (non-inverting) and ADCIN15 (inverting)<br>Fh ADCIN14 (non-inverting) and ADCIN15 (inverting)<br>Reset type: SYSRSn |
| 14-9 | RESERVED | R | 0h | Reserved |
| 8-0 | ACQPS | R/W | 0h | SOC0 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration.<br>000h Sample window is 1 system clock cycle wide<br>001h Sample window is 2 system clock cycles wide<br>002h Sample window is 3 system clock cycles wide<br>...<br>1FFh Sample window is 512 system clock cycles wide<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.18 ADCSOC1CTL Register (Offset = 12h) [Reset = 00000000h]

ADCSOC1CTL is shown in Figure 11-63 and described in Table 11-56.

Return to the Summary Table.

ADC SOC1 Control Register

**Figure 11-63. ADCSOC1CTL Register**

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

**Table 11-56. ADCSOC1CTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-25 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-56. ADCSOC1CTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 24-20 | TRIGSEL | R/W | 0h | SOC1 Trigger Source Select. Along with the SOC1 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC1 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.<br>Note: SOCFRC1 register can always be used to software trigger SOCs in addition to any hardware trigger configuration.<br>00h ADCTRIG0 - Software only<br>01h ADCTRIG1 - CPU1 Timer 0, TINT0n<br>02h ADCTRIG2 - CPU1 Timer 1, TINT1n<br>03h ADCTRIG3 - CPU1 Timer 2, TINT2n<br>04h ADCTRIG4 - GPIO, ADCEXTSOC<br>05h ADCTRIG5 - ePWM1, ADCSOCA<br>06h ADCTRIG6 - ePWM1, ADCSOCB<br>07h ADCTRIG7 - ePWM2, ADCSOCA<br>08h ADCTRIG8 - ePWM2, ADCSOCB<br>09h ADCTRIG9 - ePWM3, ADCSOCA<br>0Ah ADCTRIG10 - ePWM3, ADCSOCB<br>0Bh ADCTRIG11 - ePWM4, ADCSOCA<br>0Ch ADCTRIG12 - ePWM4, ADCSOCB<br>0Dh ADCTRIG13 - ePWM5, ADCSOCA<br>0Eh ADCTRIG14 - ePWM5, ADCSOCB<br>0Fh ADCTRIG15 - ePWM6, ADCSOCA<br>10h ADCTRIG16 - ePWM6, ADCSOCB<br>11h ADCTRIG17 - ePWM7, ADCSOCA<br>12h ADCTRIG18 - ePWM7, ADCSOCB<br>13h ADCTRIG19 - ePWM8, ADCSOCA<br>14h ADCTRIG20 - ePWM8, ADCSOCB<br>15h ADCTRIG21 - ePWM9, ADCSOCA<br>16h ADCTRIG22 - ePWM9, ADCSOCB<br>17h ADCTRIG23 - ePWM10, ADCSOCA<br>18h ADCTRIG24 - ePWM10, ADCSOCB<br>19h ADCTRIG25 - ePWM11, ADCSOCA<br>1Ah ADCTRIG26 - ePWM11, ADCSOCB<br>1Bh ADCTRIG27 - ePWM12, ADCSOCA<br>1Ch ADCTRIG28 - ePWM12, ADCSOCB<br>1Dh ADCTRIG29 - CPU2 Timer 0, TINT0n<br>1Eh ADCTRIG30 - CPU2 Timer 1, TINT1n<br>1Fh ADCTRIG31 - CPU2 Timer 2, TINT2n<br>Reset type: SYSRSn |
| 19 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-56. ADCSOC1CTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 18-15 | CHSEL | R/W | 0h | SOC1 Channel Select. Selects the channel to be converted when SOC1 is received by the ADC.<br>Single-ended Signaling Mode (SIGNALMODE = 0):<br>0h ADCIN0<br>1h ADCIN1<br>2h ADCIN2<br>3h ADCIN3<br>4h ADCIN4<br>5h ADCIN5<br>6h ADCIN6<br>7h ADCIN7<br>8h ADCIN8<br>9h ADCIN9<br>Ah ADCIN10<br>Bh ADCIN11<br>Ch ADCIN12<br>Dh ADCIN13<br>Eh ADCIN14<br>Fh ADCIN15<br>Differential Signaling Mode (SIGNALMODE = 1):<br>0h ADCIN0 (non-inverting) and ADCIN1 (inverting)<br>1h ADCIN0 (non-inverting) and ADCIN1 (inverting)<br>2h ADCIN2 (non-inverting) and ADCIN3 (inverting)<br>3h ADCIN2 (non-inverting) and ADCIN3 (inverting)<br>4h ADCIN4 (non-inverting) and ADCIN5 (inverting)<br>5h ADCIN4 (non-inverting) and ADCIN5 (inverting)<br>6h ADCIN6 (non-inverting) and ADCIN7 (inverting)<br>7h ADCIN6 (non-inverting) and ADCIN7 (inverting)<br>8h ADCIN8 (non-inverting) and ADCIN9 (inverting)<br>9h ADCIN8 (non-inverting) and ADCIN9 (inverting)<br>Ah ADCIN10 (non-inverting) and ADCIN11 (inverting)<br>Bh ADCIN10 (non-inverting) and ADCIN11 (inverting)<br>Ch ADCIN12 (non-inverting) and ADCIN13 (inverting)<br>Dh ADCIN12 (non-inverting) and ADCIN13 (inverting)<br>Eh ADCIN14 (non-inverting) and ADCIN15 (inverting)<br>Fh ADCIN14 (non-inverting) and ADCIN15 (inverting)<br>Reset type: SYSRSn |
| 14-9 | RESERVED | R | 0h | Reserved |
| 8-0 | ACQPS | R/W | 0h | SOC1 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration.<br>000h Sample window is 1 system clock cycle wide<br>001h Sample window is 2 system clock cycles wide<br>002h Sample window is 3 system clock cycles wide<br>...<br>1FFh Sample window is 512 system clock cycles wide<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.19 ADCSOC2CTL Register (Offset = 14h) [Reset = 00000000h]

ADCSOC2CTL is shown in Figure 11-64 and described in Table 11-57.

Return to the Summary Table.

ADC SOC2 Control Register

**Figure 11-64. ADCSOC2CTL Register**

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

**Table 11-57. ADCSOC2CTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-25 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-57. ADCSOC2CTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 24-20 | TRIGSEL | R/W | 0h | SOC2 Trigger Source Select. Along with the SOC2 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC2 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.<br>Note: SOCFRC1 register can always be used to software trigger SOCs in addition to any hardware trigger configuration.<br>00h ADCTRIG0 - Software only<br>01h ADCTRIG1 - CPU1 Timer 0, TINT0n<br>02h ADCTRIG2 - CPU1 Timer 1, TINT1n<br>03h ADCTRIG3 - CPU1 Timer 2, TINT2n<br>04h ADCTRIG4 - GPIO, ADCEXTSOC<br>05h ADCTRIG5 - ePWM1, ADCSOCA<br>06h ADCTRIG6 - ePWM1, ADCSOCB<br>07h ADCTRIG7 - ePWM2, ADCSOCA<br>08h ADCTRIG8 - ePWM2, ADCSOCB<br>09h ADCTRIG9 - ePWM3, ADCSOCA<br>0Ah ADCTRIG10 - ePWM3, ADCSOCB<br>0Bh ADCTRIG11 - ePWM4, ADCSOCA<br>0Ch ADCTRIG12 - ePWM4, ADCSOCB<br>0Dh ADCTRIG13 - ePWM5, ADCSOCA<br>0Eh ADCTRIG14 - ePWM5, ADCSOCB<br>0Fh ADCTRIG15 - ePWM6, ADCSOCA<br>10h ADCTRIG16 - ePWM6, ADCSOCB<br>11h ADCTRIG17 - ePWM7, ADCSOCA<br>12h ADCTRIG18 - ePWM7, ADCSOCB<br>13h ADCTRIG19 - ePWM8, ADCSOCA<br>14h ADCTRIG20 - ePWM8, ADCSOCB<br>15h ADCTRIG21 - ePWM9, ADCSOCA<br>16h ADCTRIG22 - ePWM9, ADCSOCB<br>17h ADCTRIG23 - ePWM10, ADCSOCA<br>18h ADCTRIG24 - ePWM10, ADCSOCB<br>19h ADCTRIG25 - ePWM11, ADCSOCA<br>1Ah ADCTRIG26 - ePWM11, ADCSOCB<br>1Bh ADCTRIG27 - ePWM12, ADCSOCA<br>1Ch ADCTRIG28 - ePWM12, ADCSOCB<br>1Dh ADCTRIG29 - CPU2 Timer 0, TINT0n<br>1Eh ADCTRIG30 - CPU2 Timer 1, TINT1n<br>1Fh ADCTRIG31 - CPU2 Timer 2, TINT2n<br>Reset type: SYSRSn |
| 19 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-57. ADCSOC2CTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 18-15 | CHSEL | R/W | 0h | SOC2 Channel Select. Selects the channel to be converted when SOC2 is received by the ADC.<br>Single-ended Signaling Mode (SIGNALMODE = 0):<br>0h ADCIN0<br>1h ADCIN1<br>2h ADCIN2<br>3h ADCIN3<br>4h ADCIN4<br>5h ADCIN5<br>6h ADCIN6<br>7h ADCIN7<br>8h ADCIN8<br>9h ADCIN9<br>Ah ADCIN10<br>Bh ADCIN11<br>Ch ADCIN12<br>Dh ADCIN13<br>Eh ADCIN14<br>Fh ADCIN15<br>Differential Signaling Mode (SIGNALMODE = 1):<br>0h ADCIN0 (non-inverting) and ADCIN1 (inverting)<br>1h ADCIN0 (non-inverting) and ADCIN1 (inverting)<br>2h ADCIN2 (non-inverting) and ADCIN3 (inverting)<br>3h ADCIN2 (non-inverting) and ADCIN3 (inverting)<br>4h ADCIN4 (non-inverting) and ADCIN5 (inverting)<br>5h ADCIN4 (non-inverting) and ADCIN5 (inverting)<br>6h ADCIN6 (non-inverting) and ADCIN7 (inverting)<br>7h ADCIN6 (non-inverting) and ADCIN7 (inverting)<br>8h ADCIN8 (non-inverting) and ADCIN9 (inverting)<br>9h ADCIN8 (non-inverting) and ADCIN9 (inverting)<br>Ah ADCIN10 (non-inverting) and ADCIN11 (inverting)<br>Bh ADCIN10 (non-inverting) and ADCIN11 (inverting)<br>Ch ADCIN12 (non-inverting) and ADCIN13 (inverting)<br>Dh ADCIN12 (non-inverting) and ADCIN13 (inverting)<br>Eh ADCIN14 (non-inverting) and ADCIN15 (inverting)<br>Fh ADCIN14 (non-inverting) and ADCIN15 (inverting)<br>Reset type: SYSRSn |
| 14-9 | RESERVED | R | 0h | Reserved |
| 8-0 | ACQPS | R/W | 0h | SOC2 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration.<br>000h Sample window is 1 system clock cycle wide<br>001h Sample window is 2 system clock cycles wide<br>002h Sample window is 3 system clock cycles wide<br>...<br>1FFh Sample window is 512 system clock cycles wide<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.20 ADCSOC3CTL Register (Offset = 16h) [Reset = 00000000h]

ADCSOC3CTL is shown in Figure 11-65 and described in Table 11-58.

Return to the Summary Table.

ADC SOC3 Control Register

**Figure 11-65. ADCSOC3CTL Register**

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

**Table 11-58. ADCSOC3CTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-25 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-58. ADCSOC3CTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 24-20 | TRIGSEL | R/W | 0h | SOC3 Trigger Source Select. Along with the SOC3 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC3 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.<br>Note: SOCFRC1 register can always be used to software trigger SOCs in addition to any hardware trigger configuration.<br>00h ADCTRIG0 - Software only<br>01h ADCTRIG1 - CPU1 Timer 0, TINT0n<br>02h ADCTRIG2 - CPU1 Timer 1, TINT1n<br>03h ADCTRIG3 - CPU1 Timer 2, TINT2n<br>04h ADCTRIG4 - GPIO, ADCEXTSOC<br>05h ADCTRIG5 - ePWM1, ADCSOCA<br>06h ADCTRIG6 - ePWM1, ADCSOCB<br>07h ADCTRIG7 - ePWM2, ADCSOCA<br>08h ADCTRIG8 - ePWM2, ADCSOCB<br>09h ADCTRIG9 - ePWM3, ADCSOCA<br>0Ah ADCTRIG10 - ePWM3, ADCSOCB<br>0Bh ADCTRIG11 - ePWM4, ADCSOCA<br>0Ch ADCTRIG12 - ePWM4, ADCSOCB<br>0Dh ADCTRIG13 - ePWM5, ADCSOCA<br>0Eh ADCTRIG14 - ePWM5, ADCSOCB<br>0Fh ADCTRIG15 - ePWM6, ADCSOCA<br>10h ADCTRIG16 - ePWM6, ADCSOCB<br>11h ADCTRIG17 - ePWM7, ADCSOCA<br>12h ADCTRIG18 - ePWM7, ADCSOCB<br>13h ADCTRIG19 - ePWM8, ADCSOCA<br>14h ADCTRIG20 - ePWM8, ADCSOCB<br>15h ADCTRIG21 - ePWM9, ADCSOCA<br>16h ADCTRIG22 - ePWM9, ADCSOCB<br>17h ADCTRIG23 - ePWM10, ADCSOCA<br>18h ADCTRIG24 - ePWM10, ADCSOCB<br>19h ADCTRIG25 - ePWM11, ADCSOCA<br>1Ah ADCTRIG26 - ePWM11, ADCSOCB<br>1Bh ADCTRIG27 - ePWM12, ADCSOCA<br>1Ch ADCTRIG28 - ePWM12, ADCSOCB<br>1Dh ADCTRIG29 - CPU2 Timer 0, TINT0n<br>1Eh ADCTRIG30 - CPU2 Timer 1, TINT1n<br>1Fh ADCTRIG31 - CPU2 Timer 2, TINT2n<br>Reset type: SYSRSn |
| 19 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-58. ADCSOC3CTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 18-15 | CHSEL | R/W | 0h | SOC3 Channel Select. Selects the channel to be converted when SOC3 is received by the ADC.<br>Single-ended Signaling Mode (SIGNALMODE = 0):<br>0h ADCIN0<br>1h ADCIN1<br>2h ADCIN2<br>3h ADCIN3<br>4h ADCIN4<br>5h ADCIN5<br>6h ADCIN6<br>7h ADCIN7<br>8h ADCIN8<br>9h ADCIN9<br>Ah ADCIN10<br>Bh ADCIN11<br>Ch ADCIN12<br>Dh ADCIN13<br>Eh ADCIN14<br>Fh ADCIN15<br>Differential Signaling Mode (SIGNALMODE = 1):<br>0h ADCIN0 (non-inverting) and ADCIN1 (inverting)<br>1h ADCIN0 (non-inverting) and ADCIN1 (inverting)<br>2h ADCIN2 (non-inverting) and ADCIN3 (inverting)<br>3h ADCIN2 (non-inverting) and ADCIN3 (inverting)<br>4h ADCIN4 (non-inverting) and ADCIN5 (inverting)<br>5h ADCIN4 (non-inverting) and ADCIN5 (inverting)<br>6h ADCIN6 (non-inverting) and ADCIN7 (inverting)<br>7h ADCIN6 (non-inverting) and ADCIN7 (inverting)<br>8h ADCIN8 (non-inverting) and ADCIN9 (inverting)<br>9h ADCIN8 (non-inverting) and ADCIN9 (inverting)<br>Ah ADCIN10 (non-inverting) and ADCIN11 (inverting)<br>Bh ADCIN10 (non-inverting) and ADCIN11 (inverting)<br>Ch ADCIN12 (non-inverting) and ADCIN13 (inverting)<br>Dh ADCIN12 (non-inverting) and ADCIN13 (inverting)<br>Eh ADCIN14 (non-inverting) and ADCIN15 (inverting)<br>Fh ADCIN14 (non-inverting) and ADCIN15 (inverting)<br>Reset type: SYSRSn |
| 14-9 | RESERVED | R | 0h | Reserved |
| 8-0 | ACQPS | R/W | 0h | SOC3 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration.<br>000h Sample window is 1 system clock cycle wide<br>001h Sample window is 2 system clock cycles wide<br>002h Sample window is 3 system clock cycles wide<br>...<br>1FFh Sample window is 512 system clock cycles wide<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.21 ADCSOC4CTL Register (Offset = 18h) [Reset = 00000000h]

ADCSOC4CTL is shown in Figure 11-66 and described in Table 11-59.

Return to the Summary Table.

ADC SOC4 Control Register

**Figure 11-66. ADCSOC4CTL Register**

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

**Table 11-59. ADCSOC4CTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-25 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-59. ADCSOC4CTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 24-20 | TRIGSEL | R/W | 0h | SOC4 Trigger Source Select. Along with the SOC4 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC4 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.<br>Note: SOCFRC1 register can always be used to software trigger SOCs in addition to any hardware trigger configuration.<br>00h ADCTRIG0 - Software only<br>01h ADCTRIG1 - CPU1 Timer 0, TINT0n<br>02h ADCTRIG2 - CPU1 Timer 1, TINT1n<br>03h ADCTRIG3 - CPU1 Timer 2, TINT2n<br>04h ADCTRIG4 - GPIO, ADCEXTSOC<br>05h ADCTRIG5 - ePWM1, ADCSOCA<br>06h ADCTRIG6 - ePWM1, ADCSOCB<br>07h ADCTRIG7 - ePWM2, ADCSOCA<br>08h ADCTRIG8 - ePWM2, ADCSOCB<br>09h ADCTRIG9 - ePWM3, ADCSOCA<br>0Ah ADCTRIG10 - ePWM3, ADCSOCB<br>0Bh ADCTRIG11 - ePWM4, ADCSOCA<br>0Ch ADCTRIG12 - ePWM4, ADCSOCB<br>0Dh ADCTRIG13 - ePWM5, ADCSOCA<br>0Eh ADCTRIG14 - ePWM5, ADCSOCB<br>0Fh ADCTRIG15 - ePWM6, ADCSOCA<br>10h ADCTRIG16 - ePWM6, ADCSOCB<br>11h ADCTRIG17 - ePWM7, ADCSOCA<br>12h ADCTRIG18 - ePWM7, ADCSOCB<br>13h ADCTRIG19 - ePWM8, ADCSOCA<br>14h ADCTRIG20 - ePWM8, ADCSOCB<br>15h ADCTRIG21 - ePWM9, ADCSOCA<br>16h ADCTRIG22 - ePWM9, ADCSOCB<br>17h ADCTRIG23 - ePWM10, ADCSOCA<br>18h ADCTRIG24 - ePWM10, ADCSOCB<br>19h ADCTRIG25 - ePWM11, ADCSOCA<br>1Ah ADCTRIG26 - ePWM11, ADCSOCB<br>1Bh ADCTRIG27 - ePWM12, ADCSOCA<br>1Ch ADCTRIG28 - ePWM12, ADCSOCB<br>1Dh ADCTRIG29 - CPU2 Timer 0, TINT0n<br>1Eh ADCTRIG30 - CPU2 Timer 1, TINT1n<br>1Fh ADCTRIG31 - CPU2 Timer 2, TINT2n<br>Reset type: SYSRSn |
| 19 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-59. ADCSOC4CTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 18-15 | CHSEL | R/W | 0h | SOC4 Channel Select. Selects the channel to be converted when SOC4 is received by the ADC.<br>Single-ended Signaling Mode (SIGNALMODE = 0):<br>0h ADCIN0<br>1h ADCIN1<br>2h ADCIN2<br>3h ADCIN3<br>4h ADCIN4<br>5h ADCIN5<br>6h ADCIN6<br>7h ADCIN7<br>8h ADCIN8<br>9h ADCIN9<br>Ah ADCIN10<br>Bh ADCIN11<br>Ch ADCIN12<br>Dh ADCIN13<br>Eh ADCIN14<br>Fh ADCIN15<br>Differential Signaling Mode (SIGNALMODE = 1):<br>0h ADCIN0 (non-inverting) and ADCIN1 (inverting)<br>1h ADCIN0 (non-inverting) and ADCIN1 (inverting)<br>2h ADCIN2 (non-inverting) and ADCIN3 (inverting)<br>3h ADCIN2 (non-inverting) and ADCIN3 (inverting)<br>4h ADCIN4 (non-inverting) and ADCIN5 (inverting)<br>5h ADCIN4 (non-inverting) and ADCIN5 (inverting)<br>6h ADCIN6 (non-inverting) and ADCIN7 (inverting)<br>7h ADCIN6 (non-inverting) and ADCIN7 (inverting)<br>8h ADCIN8 (non-inverting) and ADCIN9 (inverting)<br>9h ADCIN8 (non-inverting) and ADCIN9 (inverting)<br>Ah ADCIN10 (non-inverting) and ADCIN11 (inverting)<br>Bh ADCIN10 (non-inverting) and ADCIN11 (inverting)<br>Ch ADCIN12 (non-inverting) and ADCIN13 (inverting)<br>Dh ADCIN12 (non-inverting) and ADCIN13 (inverting)<br>Eh ADCIN14 (non-inverting) and ADCIN15 (inverting)<br>Fh ADCIN14 (non-inverting) and ADCIN15 (inverting)<br>Reset type: SYSRSn |
| 14-9 | RESERVED | R | 0h | Reserved |
| 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.<br>000h Sample window is 1 system clock cycle wide<br>001h Sample window is 2 system clock cycles wide<br>002h Sample window is 3 system clock cycles wide<br>...<br>1FFh Sample window is 512 system clock cycles wide<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.22 ADCSOC5CTL Register (Offset = 1Ah) [Reset = 00000000h]

ADCSOC5CTL is shown in Figure 11-67 and described in Table 11-60.

Return to the Summary Table.

ADC SOC5 Control Register

**Figure 11-67. ADCSOC5CTL Register**

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

**Table 11-60. ADCSOC5CTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-25 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-60. ADCSOC5CTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 24-20 | TRIGSEL | R/W | 0h | SOC5 Trigger Source Select. Along with the SOC5 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC5 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.<br>Note: SOCFRC1 register can always be used to software trigger SOCs in addition to any hardware trigger configuration.<br>00h ADCTRIG0 - Software only<br>01h ADCTRIG1 - CPU1 Timer 0, TINT0n<br>02h ADCTRIG2 - CPU1 Timer 1, TINT1n<br>03h ADCTRIG3 - CPU1 Timer 2, TINT2n<br>04h ADCTRIG4 - GPIO, ADCEXTSOC<br>05h ADCTRIG5 - ePWM1, ADCSOCA<br>06h ADCTRIG6 - ePWM1, ADCSOCB<br>07h ADCTRIG7 - ePWM2, ADCSOCA<br>08h ADCTRIG8 - ePWM2, ADCSOCB<br>09h ADCTRIG9 - ePWM3, ADCSOCA<br>0Ah ADCTRIG10 - ePWM3, ADCSOCB<br>0Bh ADCTRIG11 - ePWM4, ADCSOCA<br>0Ch ADCTRIG12 - ePWM4, ADCSOCB<br>0Dh ADCTRIG13 - ePWM5, ADCSOCA<br>0Eh ADCTRIG14 - ePWM5, ADCSOCB<br>0Fh ADCTRIG15 - ePWM6, ADCSOCA<br>10h ADCTRIG16 - ePWM6, ADCSOCB<br>11h ADCTRIG17 - ePWM7, ADCSOCA<br>12h ADCTRIG18 - ePWM7, ADCSOCB<br>13h ADCTRIG19 - ePWM8, ADCSOCA<br>14h ADCTRIG20 - ePWM8, ADCSOCB<br>15h ADCTRIG21 - ePWM9, ADCSOCA<br>16h ADCTRIG22 - ePWM9, ADCSOCB<br>17h ADCTRIG23 - ePWM10, ADCSOCA<br>18h ADCTRIG24 - ePWM10, ADCSOCB<br>19h ADCTRIG25 - ePWM11, ADCSOCA<br>1Ah ADCTRIG26 - ePWM11, ADCSOCB<br>1Bh ADCTRIG27 - ePWM12, ADCSOCA<br>1Ch ADCTRIG28 - ePWM12, ADCSOCB<br>1Dh ADCTRIG29 - CPU2 Timer 0, TINT0n<br>1Eh ADCTRIG30 - CPU2 Timer 1, TINT1n<br>1Fh ADCTRIG31 - CPU2 Timer 2, TINT2n<br>Reset type: SYSRSn |
| 19 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-60. ADCSOC5CTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 18-15 | CHSEL | R/W | 0h | SOC5 Channel Select. Selects the channel to be converted when SOC5 is received by the ADC.<br>Single-ended Signaling Mode (SIGNALMODE = 0):<br>0h ADCIN0<br>1h ADCIN1<br>2h ADCIN2<br>3h ADCIN3<br>4h ADCIN4<br>5h ADCIN5<br>6h ADCIN6<br>7h ADCIN7<br>8h ADCIN8<br>9h ADCIN9<br>Ah ADCIN10<br>Bh ADCIN11<br>Ch ADCIN12<br>Dh ADCIN13<br>Eh ADCIN14<br>Fh ADCIN15<br>Differential Signaling Mode (SIGNALMODE = 1):<br>0h ADCIN0 (non-inverting) and ADCIN1 (inverting)<br>1h ADCIN0 (non-inverting) and ADCIN1 (inverting)<br>2h ADCIN2 (non-inverting) and ADCIN3 (inverting)<br>3h ADCIN2 (non-inverting) and ADCIN3 (inverting)<br>4h ADCIN4 (non-inverting) and ADCIN5 (inverting)<br>5h ADCIN4 (non-inverting) and ADCIN5 (inverting)<br>6h ADCIN6 (non-inverting) and ADCIN7 (inverting)<br>7h ADCIN6 (non-inverting) and ADCIN7 (inverting)<br>8h ADCIN8 (non-inverting) and ADCIN9 (inverting)<br>9h ADCIN8 (non-inverting) and ADCIN9 (inverting)<br>Ah ADCIN10 (non-inverting) and ADCIN11 (inverting)<br>Bh ADCIN10 (non-inverting) and ADCIN11 (inverting)<br>Ch ADCIN12 (non-inverting) and ADCIN13 (inverting)<br>Dh ADCIN12 (non-inverting) and ADCIN13 (inverting)<br>Eh ADCIN14 (non-inverting) and ADCIN15 (inverting)<br>Fh ADCIN14 (non-inverting) and ADCIN15 (inverting)<br>Reset type: SYSRSn |
| 14-9 | RESERVED | R | 0h | Reserved |
| 8-0 | ACQPS | R/W | 0h | SOC5 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration.<br>000h Sample window is 1 system clock cycle wide<br>001h Sample window is 2 system clock cycles wide<br>002h Sample window is 3 system clock cycles wide<br>...<br>1FFh Sample window is 512 system clock cycles wide<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.23 ADCSOC6CTL Register (Offset = 1Ch) [Reset = 00000000h]

ADCSOC6CTL is shown in Figure 11-68 and described in Table 11-61.

Return to the Summary Table.

ADC SOC6 Control Register

**Figure 11-68. ADCSOC6CTL Register**

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

**Table 11-61. ADCSOC6CTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-25 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-61. ADCSOC6CTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 24-20 | TRIGSEL | R/W | 0h | SOC6 Trigger Source Select. Along with the SOC6 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC6 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.<br>Note: SOCFRC1 register can always be used to software trigger SOCs in addition to any hardware trigger configuration.<br>00h ADCTRIG0 - Software only<br>01h ADCTRIG1 - CPU1 Timer 0, TINT0n<br>02h ADCTRIG2 - CPU1 Timer 1, TINT1n<br>03h ADCTRIG3 - CPU1 Timer 2, TINT2n<br>04h ADCTRIG4 - GPIO, ADCEXTSOC<br>05h ADCTRIG5 - ePWM1, ADCSOCA<br>06h ADCTRIG6 - ePWM1, ADCSOCB<br>07h ADCTRIG7 - ePWM2, ADCSOCA<br>08h ADCTRIG8 - ePWM2, ADCSOCB<br>09h ADCTRIG9 - ePWM3, ADCSOCA<br>0Ah ADCTRIG10 - ePWM3, ADCSOCB<br>0Bh ADCTRIG11 - ePWM4, ADCSOCA<br>0Ch ADCTRIG12 - ePWM4, ADCSOCB<br>0Dh ADCTRIG13 - ePWM5, ADCSOCA<br>0Eh ADCTRIG14 - ePWM5, ADCSOCB<br>0Fh ADCTRIG15 - ePWM6, ADCSOCA<br>10h ADCTRIG16 - ePWM6, ADCSOCB<br>11h ADCTRIG17 - ePWM7, ADCSOCA<br>12h ADCTRIG18 - ePWM7, ADCSOCB<br>13h ADCTRIG19 - ePWM8, ADCSOCA<br>14h ADCTRIG20 - ePWM8, ADCSOCB<br>15h ADCTRIG21 - ePWM9, ADCSOCA<br>16h ADCTRIG22 - ePWM9, ADCSOCB<br>17h ADCTRIG23 - ePWM10, ADCSOCA<br>18h ADCTRIG24 - ePWM10, ADCSOCB<br>19h ADCTRIG25 - ePWM11, ADCSOCA<br>1Ah ADCTRIG26 - ePWM11, ADCSOCB<br>1Bh ADCTRIG27 - ePWM12, ADCSOCA<br>1Ch ADCTRIG28 - ePWM12, ADCSOCB<br>1Dh ADCTRIG29 - CPU2 Timer 0, TINT0n<br>1Eh ADCTRIG30 - CPU2 Timer 1, TINT1n<br>1Fh ADCTRIG31 - CPU2 Timer 2, TINT2n<br>Reset type: SYSRSn |
| 19 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-61. ADCSOC6CTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 18-15 | CHSEL | R/W | 0h | SOC6 Channel Select. Selects the channel to be converted when SOC6 is received by the ADC.<br>Single-ended Signaling Mode (SIGNALMODE = 0):<br>0h ADCIN0<br>1h ADCIN1<br>2h ADCIN2<br>3h ADCIN3<br>4h ADCIN4<br>5h ADCIN5<br>6h ADCIN6<br>7h ADCIN7<br>8h ADCIN8<br>9h ADCIN9<br>Ah ADCIN10<br>Bh ADCIN11<br>Ch ADCIN12<br>Dh ADCIN13<br>Eh ADCIN14<br>Fh ADCIN15<br>Differential Signaling Mode (SIGNALMODE = 1):<br>0h ADCIN0 (non-inverting) and ADCIN1 (inverting)<br>1h ADCIN0 (non-inverting) and ADCIN1 (inverting)<br>2h ADCIN2 (non-inverting) and ADCIN3 (inverting)<br>3h ADCIN2 (non-inverting) and ADCIN3 (inverting)<br>4h ADCIN4 (non-inverting) and ADCIN5 (inverting)<br>5h ADCIN4 (non-inverting) and ADCIN5 (inverting)<br>6h ADCIN6 (non-inverting) and ADCIN7 (inverting)<br>7h ADCIN6 (non-inverting) and ADCIN7 (inverting)<br>8h ADCIN8 (non-inverting) and ADCIN9 (inverting)<br>9h ADCIN8 (non-inverting) and ADCIN9 (inverting)<br>Ah ADCIN10 (non-inverting) and ADCIN11 (inverting)<br>Bh ADCIN10 (non-inverting) and ADCIN11 (inverting)<br>Ch ADCIN12 (non-inverting) and ADCIN13 (inverting)<br>Dh ADCIN12 (non-inverting) and ADCIN13 (inverting)<br>Eh ADCIN14 (non-inverting) and ADCIN15 (inverting)<br>Fh ADCIN14 (non-inverting) and ADCIN15 (inverting)<br>Reset type: SYSRSn |
| 14-9 | RESERVED | R | 0h | Reserved |
| 8-0 | ACQPS | R/W | 0h | SOC6 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration.<br>000h Sample window is 1 system clock cycle wide<br>001h Sample window is 2 system clock cycles wide<br>002h Sample window is 3 system clock cycles wide<br>...<br>1FFh Sample window is 512 system clock cycles wide<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.24 ADCSOC7CTL Register (Offset = 1Eh) [Reset = 00000000h]

ADCSOC7CTL is shown in Figure 11-69 and described in Table 11-62.

Return to the Summary Table.

ADC SOC7 Control Register

**Figure 11-69. ADCSOC7CTL Register**

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

**Table 11-62. ADCSOC7CTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-25 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-62. ADCSOC7CTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 24-20 | TRIGSEL | R/W | 0h | SOC7 Trigger Source Select. Along with the SOC7 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC7 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.<br>Note: SOCFRC1 register can always be used to software trigger SOCs in addition to any hardware trigger configuration.<br>00h ADCTRIG0 - Software only<br>01h ADCTRIG1 - CPU1 Timer 0, TINT0n<br>02h ADCTRIG2 - CPU1 Timer 1, TINT1n<br>03h ADCTRIG3 - CPU1 Timer 2, TINT2n<br>04h ADCTRIG4 - GPIO, ADCEXTSOC<br>05h ADCTRIG5 - ePWM1, ADCSOCA<br>06h ADCTRIG6 - ePWM1, ADCSOCB<br>07h ADCTRIG7 - ePWM2, ADCSOCA<br>08h ADCTRIG8 - ePWM2, ADCSOCB<br>09h ADCTRIG9 - ePWM3, ADCSOCA<br>0Ah ADCTRIG10 - ePWM3, ADCSOCB<br>0Bh ADCTRIG11 - ePWM4, ADCSOCA<br>0Ch ADCTRIG12 - ePWM4, ADCSOCB<br>0Dh ADCTRIG13 - ePWM5, ADCSOCA<br>0Eh ADCTRIG14 - ePWM5, ADCSOCB<br>0Fh ADCTRIG15 - ePWM6, ADCSOCA<br>10h ADCTRIG16 - ePWM6, ADCSOCB<br>11h ADCTRIG17 - ePWM7, ADCSOCA<br>12h ADCTRIG18 - ePWM7, ADCSOCB<br>13h ADCTRIG19 - ePWM8, ADCSOCA<br>14h ADCTRIG20 - ePWM8, ADCSOCB<br>15h ADCTRIG21 - ePWM9, ADCSOCA<br>16h ADCTRIG22 - ePWM9, ADCSOCB<br>17h ADCTRIG23 - ePWM10, ADCSOCA<br>18h ADCTRIG24 - ePWM10, ADCSOCB<br>19h ADCTRIG25 - ePWM11, ADCSOCA<br>1Ah ADCTRIG26 - ePWM11, ADCSOCB<br>1Bh ADCTRIG27 - ePWM12, ADCSOCA<br>1Ch ADCTRIG28 - ePWM12, ADCSOCB<br>1Dh ADCTRIG29 - CPU2 Timer 0, TINT0n<br>1Eh ADCTRIG30 - CPU2 Timer 1, TINT1n<br>1Fh ADCTRIG31 - CPU2 Timer 2, TINT2n<br>Reset type: SYSRSn |
| 19 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-62. ADCSOC7CTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 18-15 | CHSEL | R/W | 0h | SOC7 Channel Select. Selects the channel to be converted when SOC7 is received by the ADC.<br>Single-ended Signaling Mode (SIGNALMODE = 0):<br>0h ADCIN0<br>1h ADCIN1<br>2h ADCIN2<br>3h ADCIN3<br>4h ADCIN4<br>5h ADCIN5<br>6h ADCIN6<br>7h ADCIN7<br>8h ADCIN8<br>9h ADCIN9<br>Ah ADCIN10<br>Bh ADCIN11<br>Ch ADCIN12<br>Dh ADCIN13<br>Eh ADCIN14<br>Fh ADCIN15<br>Differential Signaling Mode (SIGNALMODE = 1):<br>0h ADCIN0 (non-inverting) and ADCIN1 (inverting)<br>1h ADCIN0 (non-inverting) and ADCIN1 (inverting)<br>2h ADCIN2 (non-inverting) and ADCIN3 (inverting)<br>3h ADCIN2 (non-inverting) and ADCIN3 (inverting)<br>4h ADCIN4 (non-inverting) and ADCIN5 (inverting)<br>5h ADCIN4 (non-inverting) and ADCIN5 (inverting)<br>6h ADCIN6 (non-inverting) and ADCIN7 (inverting)<br>7h ADCIN6 (non-inverting) and ADCIN7 (inverting)<br>8h ADCIN8 (non-inverting) and ADCIN9 (inverting)<br>9h ADCIN8 (non-inverting) and ADCIN9 (inverting)<br>Ah ADCIN10 (non-inverting) and ADCIN11 (inverting)<br>Bh ADCIN10 (non-inverting) and ADCIN11 (inverting)<br>Ch ADCIN12 (non-inverting) and ADCIN13 (inverting)<br>Dh ADCIN12 (non-inverting) and ADCIN13 (inverting)<br>Eh ADCIN14 (non-inverting) and ADCIN15 (inverting)<br>Fh ADCIN14 (non-inverting) and ADCIN15 (inverting)<br>Reset type: SYSRSn |
| 14-9 | RESERVED | R | 0h | Reserved |
| 8-0 | ACQPS | R/W | 0h | SOC7 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration.<br>000h Sample window is 1 system clock cycle wide<br>001h Sample window is 2 system clock cycles wide<br>002h Sample window is 3 system clock cycles wide<br>...<br>1FFh Sample window is 512 system clock cycles wide<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.25 ADCSOC8CTL Register (Offset = 20h) [Reset = 00000000h]

ADCSOC8CTL is shown in Figure 11-70 and described in Table 11-63.

Return to the Summary Table.

ADC SOC8 Control Register

**Figure 11-70. ADCSOC8CTL Register**

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

**Table 11-63. ADCSOC8CTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-25 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-63. ADCSOC8CTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 24-20 | TRIGSEL | R/W | 0h | SOC8 Trigger Source Select. Along with the SOC8 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC8 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.<br>Note: SOCFRC1 register can always be used to software trigger SOCs in addition to any hardware trigger configuration.<br>00h ADCTRIG0 - Software only<br>01h ADCTRIG1 - CPU1 Timer 0, TINT0n<br>02h ADCTRIG2 - CPU1 Timer 1, TINT1n<br>03h ADCTRIG3 - CPU1 Timer 2, TINT2n<br>04h ADCTRIG4 - GPIO, ADCEXTSOC<br>05h ADCTRIG5 - ePWM1, ADCSOCA<br>06h ADCTRIG6 - ePWM1, ADCSOCB<br>07h ADCTRIG7 - ePWM2, ADCSOCA<br>08h ADCTRIG8 - ePWM2, ADCSOCB<br>09h ADCTRIG9 - ePWM3, ADCSOCA<br>0Ah ADCTRIG10 - ePWM3, ADCSOCB<br>0Bh ADCTRIG11 - ePWM4, ADCSOCA<br>0Ch ADCTRIG12 - ePWM4, ADCSOCB<br>0Dh ADCTRIG13 - ePWM5, ADCSOCA<br>0Eh ADCTRIG14 - ePWM5, ADCSOCB<br>0Fh ADCTRIG15 - ePWM6, ADCSOCA<br>10h ADCTRIG16 - ePWM6, ADCSOCB<br>11h ADCTRIG17 - ePWM7, ADCSOCA<br>12h ADCTRIG18 - ePWM7, ADCSOCB<br>13h ADCTRIG19 - ePWM8, ADCSOCA<br>14h ADCTRIG20 - ePWM8, ADCSOCB<br>15h ADCTRIG21 - ePWM9, ADCSOCA<br>16h ADCTRIG22 - ePWM9, ADCSOCB<br>17h ADCTRIG23 - ePWM10, ADCSOCA<br>18h ADCTRIG24 - ePWM10, ADCSOCB<br>19h ADCTRIG25 - ePWM11, ADCSOCA<br>1Ah ADCTRIG26 - ePWM11, ADCSOCB<br>1Bh ADCTRIG27 - ePWM12, ADCSOCA<br>1Ch ADCTRIG28 - ePWM12, ADCSOCB<br>1Dh ADCTRIG29 - CPU2 Timer 0, TINT0n<br>1Eh ADCTRIG30 - CPU2 Timer 1, TINT1n<br>1Fh ADCTRIG31 - CPU2 Timer 2, TINT2n<br>Reset type: SYSRSn |
| 19 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-63. ADCSOC8CTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 18-15 | CHSEL | R/W | 0h | SOC8 Channel Select. Selects the channel to be converted when SOC8 is received by the ADC.<br>Single-ended Signaling Mode (SIGNALMODE = 0):<br>0h ADCIN0<br>1h ADCIN1<br>2h ADCIN2<br>3h ADCIN3<br>4h ADCIN4<br>5h ADCIN5<br>6h ADCIN6<br>7h ADCIN7<br>8h ADCIN8<br>9h ADCIN9<br>Ah ADCIN10<br>Bh ADCIN11<br>Ch ADCIN12<br>Dh ADCIN13<br>Eh ADCIN14<br>Fh ADCIN15<br>Differential Signaling Mode (SIGNALMODE = 1):<br>0h ADCIN0 (non-inverting) and ADCIN1 (inverting)<br>1h ADCIN0 (non-inverting) and ADCIN1 (inverting)<br>2h ADCIN2 (non-inverting) and ADCIN3 (inverting)<br>3h ADCIN2 (non-inverting) and ADCIN3 (inverting)<br>4h ADCIN4 (non-inverting) and ADCIN5 (inverting)<br>5h ADCIN4 (non-inverting) and ADCIN5 (inverting)<br>6h ADCIN6 (non-inverting) and ADCIN7 (inverting)<br>7h ADCIN6 (non-inverting) and ADCIN7 (inverting)<br>8h ADCIN8 (non-inverting) and ADCIN9 (inverting)<br>9h ADCIN8 (non-inverting) and ADCIN9 (inverting)<br>Ah ADCIN10 (non-inverting) and ADCIN11 (inverting)<br>Bh ADCIN10 (non-inverting) and ADCIN11 (inverting)<br>Ch ADCIN12 (non-inverting) and ADCIN13 (inverting)<br>Dh ADCIN12 (non-inverting) and ADCIN13 (inverting)<br>Eh ADCIN14 (non-inverting) and ADCIN15 (inverting)<br>Fh ADCIN14 (non-inverting) and ADCIN15 (inverting)<br>Reset type: SYSRSn |
| 14-9 | RESERVED | R | 0h | Reserved |
| 8-0 | ACQPS | R/W | 0h | SOC8 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration.<br>000h Sample window is 1 system clock cycle wide<br>001h Sample window is 2 system clock cycles wide<br>002h Sample window is 3 system clock cycles wide<br>...<br>1FFh Sample window is 512 system clock cycles wide<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.26 ADCSOC9CTL Register (Offset = 22h) [Reset = 00000000h]

ADCSOC9CTL is shown in Figure 11-71 and described in Table 11-64.

Return to the Summary Table.

ADC SOC9 Control Register

**Figure 11-71. ADCSOC9CTL Register**

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

**Table 11-64. ADCSOC9CTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-25 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-64. ADCSOC9CTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 24-20 | TRIGSEL | R/W | 0h | SOC9 Trigger Source Select. Along with the SOC9 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC9 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.<br>Note: SOCFRC1 register can always be used to software trigger SOCs in addition to any hardware trigger configuration.<br>00h ADCTRIG0 - Software only<br>01h ADCTRIG1 - CPU1 Timer 0, TINT0n<br>02h ADCTRIG2 - CPU1 Timer 1, TINT1n<br>03h ADCTRIG3 - CPU1 Timer 2, TINT2n<br>04h ADCTRIG4 - GPIO, ADCEXTSOC<br>05h ADCTRIG5 - ePWM1, ADCSOCA<br>06h ADCTRIG6 - ePWM1, ADCSOCB<br>07h ADCTRIG7 - ePWM2, ADCSOCA<br>08h ADCTRIG8 - ePWM2, ADCSOCB<br>09h ADCTRIG9 - ePWM3, ADCSOCA<br>0Ah ADCTRIG10 - ePWM3, ADCSOCB<br>0Bh ADCTRIG11 - ePWM4, ADCSOCA<br>0Ch ADCTRIG12 - ePWM4, ADCSOCB<br>0Dh ADCTRIG13 - ePWM5, ADCSOCA<br>0Eh ADCTRIG14 - ePWM5, ADCSOCB<br>0Fh ADCTRIG15 - ePWM6, ADCSOCA<br>10h ADCTRIG16 - ePWM6, ADCSOCB<br>11h ADCTRIG17 - ePWM7, ADCSOCA<br>12h ADCTRIG18 - ePWM7, ADCSOCB<br>13h ADCTRIG19 - ePWM8, ADCSOCA<br>14h ADCTRIG20 - ePWM8, ADCSOCB<br>15h ADCTRIG21 - ePWM9, ADCSOCA<br>16h ADCTRIG22 - ePWM9, ADCSOCB<br>17h ADCTRIG23 - ePWM10, ADCSOCA<br>18h ADCTRIG24 - ePWM10, ADCSOCB<br>19h ADCTRIG25 - ePWM11, ADCSOCA<br>1Ah ADCTRIG26 - ePWM11, ADCSOCB<br>1Bh ADCTRIG27 - ePWM12, ADCSOCA<br>1Ch ADCTRIG28 - ePWM12, ADCSOCB<br>1Dh ADCTRIG29 - CPU2 Timer 0, TINT0n<br>1Eh ADCTRIG30 - CPU2 Timer 1, TINT1n<br>1Fh ADCTRIG31 - CPU2 Timer 2, TINT2n<br>Reset type: SYSRSn |
| 19 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-64. ADCSOC9CTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 18-15 | CHSEL | R/W | 0h | SOC9 Channel Select. Selects the channel to be converted when SOC9 is received by the ADC.<br>Single-ended Signaling Mode (SIGNALMODE = 0):<br>0h ADCIN0<br>1h ADCIN1<br>2h ADCIN2<br>3h ADCIN3<br>4h ADCIN4<br>5h ADCIN5<br>6h ADCIN6<br>7h ADCIN7<br>8h ADCIN8<br>9h ADCIN9<br>Ah ADCIN10<br>Bh ADCIN11<br>Ch ADCIN12<br>Dh ADCIN13<br>Eh ADCIN14<br>Fh ADCIN15<br>Differential Signaling Mode (SIGNALMODE = 1):<br>0h ADCIN0 (non-inverting) and ADCIN1 (inverting)<br>1h ADCIN0 (non-inverting) and ADCIN1 (inverting)<br>2h ADCIN2 (non-inverting) and ADCIN3 (inverting)<br>3h ADCIN2 (non-inverting) and ADCIN3 (inverting)<br>4h ADCIN4 (non-inverting) and ADCIN5 (inverting)<br>5h ADCIN4 (non-inverting) and ADCIN5 (inverting)<br>6h ADCIN6 (non-inverting) and ADCIN7 (inverting)<br>7h ADCIN6 (non-inverting) and ADCIN7 (inverting)<br>8h ADCIN8 (non-inverting) and ADCIN9 (inverting)<br>9h ADCIN8 (non-inverting) and ADCIN9 (inverting)<br>Ah ADCIN10 (non-inverting) and ADCIN11 (inverting)<br>Bh ADCIN10 (non-inverting) and ADCIN11 (inverting)<br>Ch ADCIN12 (non-inverting) and ADCIN13 (inverting)<br>Dh ADCIN12 (non-inverting) and ADCIN13 (inverting)<br>Eh ADCIN14 (non-inverting) and ADCIN15 (inverting)<br>Fh ADCIN14 (non-inverting) and ADCIN15 (inverting)<br>Reset type: SYSRSn |
| 14-9 | RESERVED | R | 0h | Reserved |
| 8-0 | ACQPS | R/W | 0h | SOC9 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration.<br>000h Sample window is 1 system clock cycle wide<br>001h Sample window is 2 system clock cycles wide<br>002h Sample window is 3 system clock cycles wide<br>...<br>1FFh Sample window is 512 system clock cycles wide<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.27 ADCSOC10CTL Register (Offset = 24h) [Reset = 00000000h]

ADCSOC10CTL is shown in Figure 11-72 and described in Table 11-65.

Return to the Summary Table.

ADC SOC10 Control Register

**Figure 11-72. ADCSOC10CTL Register**

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

**Table 11-65. ADCSOC10CTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-25 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-65. ADCSOC10CTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 24-20 | TRIGSEL | R/W | 0h | SOC10 Trigger Source Select. Along with the SOC10 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC10 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.<br>Note: SOCFRC1 register can always be used to software trigger SOCs in addition to any hardware trigger configuration.<br>00h ADCTRIG0 - Software only<br>01h ADCTRIG1 - CPU1 Timer 0, TINT0n<br>02h ADCTRIG2 - CPU1 Timer 1, TINT1n<br>03h ADCTRIG3 - CPU1 Timer 2, TINT2n<br>04h ADCTRIG4 - GPIO, ADCEXTSOC<br>05h ADCTRIG5 - ePWM1, ADCSOCA<br>06h ADCTRIG6 - ePWM1, ADCSOCB<br>07h ADCTRIG7 - ePWM2, ADCSOCA<br>08h ADCTRIG8 - ePWM2, ADCSOCB<br>09h ADCTRIG9 - ePWM3, ADCSOCA<br>0Ah ADCTRIG10 - ePWM3, ADCSOCB<br>0Bh ADCTRIG11 - ePWM4, ADCSOCA<br>0Ch ADCTRIG12 - ePWM4, ADCSOCB<br>0Dh ADCTRIG13 - ePWM5, ADCSOCA<br>0Eh ADCTRIG14 - ePWM5, ADCSOCB<br>0Fh ADCTRIG15 - ePWM6, ADCSOCA<br>10h ADCTRIG16 - ePWM6, ADCSOCB<br>11h ADCTRIG17 - ePWM7, ADCSOCA<br>12h ADCTRIG18 - ePWM7, ADCSOCB<br>13h ADCTRIG19 - ePWM8, ADCSOCA<br>14h ADCTRIG20 - ePWM8, ADCSOCB<br>15h ADCTRIG21 - ePWM9, ADCSOCA<br>16h ADCTRIG22 - ePWM9, ADCSOCB<br>17h ADCTRIG23 - ePWM10, ADCSOCA<br>18h ADCTRIG24 - ePWM10, ADCSOCB<br>19h ADCTRIG25 - ePWM11, ADCSOCA<br>1Ah ADCTRIG26 - ePWM11, ADCSOCB<br>1Bh ADCTRIG27 - ePWM12, ADCSOCA<br>1Ch ADCTRIG28 - ePWM12, ADCSOCB<br>1Dh ADCTRIG29 - CPU2 Timer 0, TINT0n<br>1Eh ADCTRIG30 - CPU2 Timer 1, TINT1n<br>1Fh ADCTRIG31 - CPU2 Timer 2, TINT2n<br>Reset type: SYSRSn |
| 19 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-65. ADCSOC10CTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 18-15 | CHSEL | R/W | 0h | SOC10 Channel Select. Selects the channel to be converted when SOC10 is received by the ADC.<br>Single-ended Signaling Mode (SIGNALMODE = 0):<br>0h ADCIN0<br>1h ADCIN1<br>2h ADCIN2<br>3h ADCIN3<br>4h ADCIN4<br>5h ADCIN5<br>6h ADCIN6<br>7h ADCIN7<br>8h ADCIN8<br>9h ADCIN9<br>Ah ADCIN10<br>Bh ADCIN11<br>Ch ADCIN12<br>Dh ADCIN13<br>Eh ADCIN14<br>Fh ADCIN15<br>Differential Signaling Mode (SIGNALMODE = 1):<br>0h ADCIN0 (non-inverting) and ADCIN1 (inverting)<br>1h ADCIN0 (non-inverting) and ADCIN1 (inverting)<br>2h ADCIN2 (non-inverting) and ADCIN3 (inverting)<br>3h ADCIN2 (non-inverting) and ADCIN3 (inverting)<br>4h ADCIN4 (non-inverting) and ADCIN5 (inverting)<br>5h ADCIN4 (non-inverting) and ADCIN5 (inverting)<br>6h ADCIN6 (non-inverting) and ADCIN7 (inverting)<br>7h ADCIN6 (non-inverting) and ADCIN7 (inverting)<br>8h ADCIN8 (non-inverting) and ADCIN9 (inverting)<br>9h ADCIN8 (non-inverting) and ADCIN9 (inverting)<br>Ah ADCIN10 (non-inverting) and ADCIN11 (inverting)<br>Bh ADCIN10 (non-inverting) and ADCIN11 (inverting)<br>Ch ADCIN12 (non-inverting) and ADCIN13 (inverting)<br>Dh ADCIN12 (non-inverting) and ADCIN13 (inverting)<br>Eh ADCIN14 (non-inverting) and ADCIN15 (inverting)<br>Fh ADCIN14 (non-inverting) and ADCIN15 (inverting)<br>Reset type: SYSRSn |
| 14-9 | RESERVED | R | 0h | Reserved |
| 8-0 | ACQPS | R/W | 0h | SOC10 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration.<br>000h Sample window is 1 system clock cycle wide<br>001h Sample window is 2 system clock cycles wide<br>002h Sample window is 3 system clock cycles wide<br>...<br>1FFh Sample window is 512 system clock cycles wide<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.28 ADCSOC11CTL Register (Offset = 26h) [Reset = 00000000h]

ADCSOC11CTL is shown in Figure 11-73 and described in Table 11-66.

Return to the Summary Table.

ADC SOC11 Control Register

**Figure 11-73. ADCSOC11CTL Register**

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

**Table 11-66. ADCSOC11CTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-25 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-66. ADCSOC11CTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 24-20 | TRIGSEL | R/W | 0h | SOC11 Trigger Source Select. Along with the SOC11 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC11 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.<br>Note: SOCFRC1 register can always be used to software trigger SOCs in addition to any hardware trigger configuration.<br>00h ADCTRIG0 - Software only<br>01h ADCTRIG1 - CPU1 Timer 0, TINT0n<br>02h ADCTRIG2 - CPU1 Timer 1, TINT1n<br>03h ADCTRIG3 - CPU1 Timer 2, TINT2n<br>04h ADCTRIG4 - GPIO, ADCEXTSOC<br>05h ADCTRIG5 - ePWM1, ADCSOCA<br>06h ADCTRIG6 - ePWM1, ADCSOCB<br>07h ADCTRIG7 - ePWM2, ADCSOCA<br>08h ADCTRIG8 - ePWM2, ADCSOCB<br>09h ADCTRIG9 - ePWM3, ADCSOCA<br>0Ah ADCTRIG10 - ePWM3, ADCSOCB<br>0Bh ADCTRIG11 - ePWM4, ADCSOCA<br>0Ch ADCTRIG12 - ePWM4, ADCSOCB<br>0Dh ADCTRIG13 - ePWM5, ADCSOCA<br>0Eh ADCTRIG14 - ePWM5, ADCSOCB<br>0Fh ADCTRIG15 - ePWM6, ADCSOCA<br>10h ADCTRIG16 - ePWM6, ADCSOCB<br>11h ADCTRIG17 - ePWM7, ADCSOCA<br>12h ADCTRIG18 - ePWM7, ADCSOCB<br>13h ADCTRIG19 - ePWM8, ADCSOCA<br>14h ADCTRIG20 - ePWM8, ADCSOCB<br>15h ADCTRIG21 - ePWM9, ADCSOCA<br>16h ADCTRIG22 - ePWM9, ADCSOCB<br>17h ADCTRIG23 - ePWM10, ADCSOCA<br>18h ADCTRIG24 - ePWM10, ADCSOCB<br>19h ADCTRIG25 - ePWM11, ADCSOCA<br>1Ah ADCTRIG26 - ePWM11, ADCSOCB<br>1Bh ADCTRIG27 - ePWM12, ADCSOCA<br>1Ch ADCTRIG28 - ePWM12, ADCSOCB<br>1Dh ADCTRIG29 - CPU2 Timer 0, TINT0n<br>1Eh ADCTRIG30 - CPU2 Timer 1, TINT1n<br>1Fh ADCTRIG31 - CPU2 Timer 2, TINT2n<br>Reset type: SYSRSn |
| 19 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-66. ADCSOC11CTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 18-15 | CHSEL | R/W | 0h | SOC11 Channel Select. Selects the channel to be converted when SOC11 is received by the ADC.<br>Single-ended Signaling Mode (SIGNALMODE = 0):<br>0h ADCIN0<br>1h ADCIN1<br>2h ADCIN2<br>3h ADCIN3<br>4h ADCIN4<br>5h ADCIN5<br>6h ADCIN6<br>7h ADCIN7<br>8h ADCIN8<br>9h ADCIN9<br>Ah ADCIN10<br>Bh ADCIN11<br>Ch ADCIN12<br>Dh ADCIN13<br>Eh ADCIN14<br>Fh ADCIN15<br>Differential Signaling Mode (SIGNALMODE = 1):<br>0h ADCIN0 (non-inverting) and ADCIN1 (inverting)<br>1h ADCIN0 (non-inverting) and ADCIN1 (inverting)<br>2h ADCIN2 (non-inverting) and ADCIN3 (inverting)<br>3h ADCIN2 (non-inverting) and ADCIN3 (inverting)<br>4h ADCIN4 (non-inverting) and ADCIN5 (inverting)<br>5h ADCIN4 (non-inverting) and ADCIN5 (inverting)<br>6h ADCIN6 (non-inverting) and ADCIN7 (inverting)<br>7h ADCIN6 (non-inverting) and ADCIN7 (inverting)<br>8h ADCIN8 (non-inverting) and ADCIN9 (inverting)<br>9h ADCIN8 (non-inverting) and ADCIN9 (inverting)<br>Ah ADCIN10 (non-inverting) and ADCIN11 (inverting)<br>Bh ADCIN10 (non-inverting) and ADCIN11 (inverting)<br>Ch ADCIN12 (non-inverting) and ADCIN13 (inverting)<br>Dh ADCIN12 (non-inverting) and ADCIN13 (inverting)<br>Eh ADCIN14 (non-inverting) and ADCIN15 (inverting)<br>Fh ADCIN14 (non-inverting) and ADCIN15 (inverting)<br>Reset type: SYSRSn |
| 14-9 | RESERVED | R | 0h | Reserved |
| 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.<br>000h Sample window is 1 system clock cycle wide<br>001h Sample window is 2 system clock cycles wide<br>002h Sample window is 3 system clock cycles wide<br>...<br>1FFh Sample window is 512 system clock cycles wide<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.29 ADCSOC12CTL Register (Offset = 28h) [Reset = 00000000h]

ADCSOC12CTL is shown in Figure 11-74 and described in Table 11-67.

Return to the Summary Table.

ADC SOC12 Control Register

**Figure 11-74. ADCSOC12CTL Register**

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

**Table 11-67. ADCSOC12CTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-25 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-67. ADCSOC12CTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 24-20 | TRIGSEL | R/W | 0h | SOC12 Trigger Source Select. Along with the SOC12 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC12 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.<br>Note: SOCFRC1 register can always be used to software trigger SOCs in addition to any hardware trigger configuration.<br>00h ADCTRIG0 - Software only<br>01h ADCTRIG1 - CPU1 Timer 0, TINT0n<br>02h ADCTRIG2 - CPU1 Timer 1, TINT1n<br>03h ADCTRIG3 - CPU1 Timer 2, TINT2n<br>04h ADCTRIG4 - GPIO, ADCEXTSOC<br>05h ADCTRIG5 - ePWM1, ADCSOCA<br>06h ADCTRIG6 - ePWM1, ADCSOCB<br>07h ADCTRIG7 - ePWM2, ADCSOCA<br>08h ADCTRIG8 - ePWM2, ADCSOCB<br>09h ADCTRIG9 - ePWM3, ADCSOCA<br>0Ah ADCTRIG10 - ePWM3, ADCSOCB<br>0Bh ADCTRIG11 - ePWM4, ADCSOCA<br>0Ch ADCTRIG12 - ePWM4, ADCSOCB<br>0Dh ADCTRIG13 - ePWM5, ADCSOCA<br>0Eh ADCTRIG14 - ePWM5, ADCSOCB<br>0Fh ADCTRIG15 - ePWM6, ADCSOCA<br>10h ADCTRIG16 - ePWM6, ADCSOCB<br>11h ADCTRIG17 - ePWM7, ADCSOCA<br>12h ADCTRIG18 - ePWM7, ADCSOCB<br>13h ADCTRIG19 - ePWM8, ADCSOCA<br>14h ADCTRIG20 - ePWM8, ADCSOCB<br>15h ADCTRIG21 - ePWM9, ADCSOCA<br>16h ADCTRIG22 - ePWM9, ADCSOCB<br>17h ADCTRIG23 - ePWM10, ADCSOCA<br>18h ADCTRIG24 - ePWM10, ADCSOCB<br>19h ADCTRIG25 - ePWM11, ADCSOCA<br>1Ah ADCTRIG26 - ePWM11, ADCSOCB<br>1Bh ADCTRIG27 - ePWM12, ADCSOCA<br>1Ch ADCTRIG28 - ePWM12, ADCSOCB<br>1Dh ADCTRIG29 - CPU2 Timer 0, TINT0n<br>1Eh ADCTRIG30 - CPU2 Timer 1, TINT1n<br>1Fh ADCTRIG31 - CPU2 Timer 2, TINT2n<br>Reset type: SYSRSn |
| 19 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-67. ADCSOC12CTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 18-15 | CHSEL | R/W | 0h | SOC12 Channel Select. Selects the channel to be converted when SOC12 is received by the ADC.<br>Single-ended Signaling Mode (SIGNALMODE = 0):<br>0h ADCIN0<br>1h ADCIN1<br>2h ADCIN2<br>3h ADCIN3<br>4h ADCIN4<br>5h ADCIN5<br>6h ADCIN6<br>7h ADCIN7<br>8h ADCIN8<br>9h ADCIN9<br>Ah ADCIN10<br>Bh ADCIN11<br>Ch ADCIN12<br>Dh ADCIN13<br>Eh ADCIN14<br>Fh ADCIN15<br>Differential Signaling Mode (SIGNALMODE = 1):<br>0h ADCIN0 (non-inverting) and ADCIN1 (inverting)<br>1h ADCIN0 (non-inverting) and ADCIN1 (inverting)<br>2h ADCIN2 (non-inverting) and ADCIN3 (inverting)<br>3h ADCIN2 (non-inverting) and ADCIN3 (inverting)<br>4h ADCIN4 (non-inverting) and ADCIN5 (inverting)<br>5h ADCIN4 (non-inverting) and ADCIN5 (inverting)<br>6h ADCIN6 (non-inverting) and ADCIN7 (inverting)<br>7h ADCIN6 (non-inverting) and ADCIN7 (inverting)<br>8h ADCIN8 (non-inverting) and ADCIN9 (inverting)<br>9h ADCIN8 (non-inverting) and ADCIN9 (inverting)<br>Ah ADCIN10 (non-inverting) and ADCIN11 (inverting)<br>Bh ADCIN10 (non-inverting) and ADCIN11 (inverting)<br>Ch ADCIN12 (non-inverting) and ADCIN13 (inverting)<br>Dh ADCIN12 (non-inverting) and ADCIN13 (inverting)<br>Eh ADCIN14 (non-inverting) and ADCIN15 (inverting)<br>Fh ADCIN14 (non-inverting) and ADCIN15 (inverting)<br>Reset type: SYSRSn |
| 14-9 | RESERVED | R | 0h | Reserved |
| 8-0 | ACQPS | R/W | 0h | SOC12 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration.<br>000h Sample window is 1 system clock cycle wide<br>001h Sample window is 2 system clock cycles wide<br>002h Sample window is 3 system clock cycles wide<br>...<br>1FFh Sample window is 512 system clock cycles wide<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.30 ADCSOC13CTL Register (Offset = 2Ah) [Reset = 00000000h]

ADCSOC13CTL is shown in Figure 11-75 and described in Table 11-68.

Return to the Summary Table.

ADC SOC13 Control Register

**Figure 11-75. ADCSOC13CTL Register**

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

**Table 11-68. ADCSOC13CTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-25 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-68. ADCSOC13CTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 24-20 | TRIGSEL | R/W | 0h | SOC13 Trigger Source Select. Along with the SOC13 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC13 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.<br>Note: SOCFRC1 register can always be used to software trigger SOCs in addition to any hardware trigger configuration.<br>00h ADCTRIG0 - Software only<br>01h ADCTRIG1 - CPU1 Timer 0, TINT0n<br>02h ADCTRIG2 - CPU1 Timer 1, TINT1n<br>03h ADCTRIG3 - CPU1 Timer 2, TINT2n<br>04h ADCTRIG4 - GPIO, ADCEXTSOC<br>05h ADCTRIG5 - ePWM1, ADCSOCA<br>06h ADCTRIG6 - ePWM1, ADCSOCB<br>07h ADCTRIG7 - ePWM2, ADCSOCA<br>08h ADCTRIG8 - ePWM2, ADCSOCB<br>09h ADCTRIG9 - ePWM3, ADCSOCA<br>0Ah ADCTRIG10 - ePWM3, ADCSOCB<br>0Bh ADCTRIG11 - ePWM4, ADCSOCA<br>0Ch ADCTRIG12 - ePWM4, ADCSOCB<br>0Dh ADCTRIG13 - ePWM5, ADCSOCA<br>0Eh ADCTRIG14 - ePWM5, ADCSOCB<br>0Fh ADCTRIG15 - ePWM6, ADCSOCA<br>10h ADCTRIG16 - ePWM6, ADCSOCB<br>11h ADCTRIG17 - ePWM7, ADCSOCA<br>12h ADCTRIG18 - ePWM7, ADCSOCB<br>13h ADCTRIG19 - ePWM8, ADCSOCA<br>14h ADCTRIG20 - ePWM8, ADCSOCB<br>15h ADCTRIG21 - ePWM9, ADCSOCA<br>16h ADCTRIG22 - ePWM9, ADCSOCB<br>17h ADCTRIG23 - ePWM10, ADCSOCA<br>18h ADCTRIG24 - ePWM10, ADCSOCB<br>19h ADCTRIG25 - ePWM11, ADCSOCA<br>1Ah ADCTRIG26 - ePWM11, ADCSOCB<br>1Bh ADCTRIG27 - ePWM12, ADCSOCA<br>1Ch ADCTRIG28 - ePWM12, ADCSOCB<br>1Dh ADCTRIG29 - CPU2 Timer 0, TINT0n<br>1Eh ADCTRIG30 - CPU2 Timer 1, TINT1n<br>1Fh ADCTRIG31 - CPU2 Timer 2, TINT2n<br>Reset type: SYSRSn |
| 19 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-68. ADCSOC13CTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 18-15 | CHSEL | R/W | 0h | SOC13 Channel Select. Selects the channel to be converted when SOC13 is received by the ADC.<br>Single-ended Signaling Mode (SIGNALMODE = 0):<br>0h ADCIN0<br>1h ADCIN1<br>2h ADCIN2<br>3h ADCIN3<br>4h ADCIN4<br>5h ADCIN5<br>6h ADCIN6<br>7h ADCIN7<br>8h ADCIN8<br>9h ADCIN9<br>Ah ADCIN10<br>Bh ADCIN11<br>Ch ADCIN12<br>Dh ADCIN13<br>Eh ADCIN14<br>Fh ADCIN15<br>Differential Signaling Mode (SIGNALMODE = 1):<br>0h ADCIN0 (non-inverting) and ADCIN1 (inverting)<br>1h ADCIN0 (non-inverting) and ADCIN1 (inverting)<br>2h ADCIN2 (non-inverting) and ADCIN3 (inverting)<br>3h ADCIN2 (non-inverting) and ADCIN3 (inverting)<br>4h ADCIN4 (non-inverting) and ADCIN5 (inverting)<br>5h ADCIN4 (non-inverting) and ADCIN5 (inverting)<br>6h ADCIN6 (non-inverting) and ADCIN7 (inverting)<br>7h ADCIN6 (non-inverting) and ADCIN7 (inverting)<br>8h ADCIN8 (non-inverting) and ADCIN9 (inverting)<br>9h ADCIN8 (non-inverting) and ADCIN9 (inverting)<br>Ah ADCIN10 (non-inverting) and ADCIN11 (inverting)<br>Bh ADCIN10 (non-inverting) and ADCIN11 (inverting)<br>Ch ADCIN12 (non-inverting) and ADCIN13 (inverting)<br>Dh ADCIN12 (non-inverting) and ADCIN13 (inverting)<br>Eh ADCIN14 (non-inverting) and ADCIN15 (inverting)<br>Fh ADCIN14 (non-inverting) and ADCIN15 (inverting)<br>Reset type: SYSRSn |
| 14-9 | RESERVED | R | 0h | Reserved |
| 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.<br>000h Sample window is 1 system clock cycle wide<br>001h Sample window is 2 system clock cycles wide<br>002h Sample window is 3 system clock cycles wide<br>...<br>1FFh Sample window is 512 system clock cycles wide<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.31 ADCSOC14CTL Register (Offset = 2Ch) [Reset = 00000000h]

ADCSOC14CTL is shown in Figure 11-76 and described in Table 11-69.

Return to the Summary Table.

ADC SOC14 Control Register

**Figure 11-76. ADCSOC14CTL Register**

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

**Table 11-69. ADCSOC14CTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-25 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-69. ADCSOC14CTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 24-20 | TRIGSEL | R/W | 0h | SOC14 Trigger Source Select. Along with the SOC14 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC14 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.<br>Note: SOCFRC1 register can always be used to software trigger SOCs in addition to any hardware trigger configuration.<br>00h ADCTRIG0 - Software only<br>01h ADCTRIG1 - CPU1 Timer 0, TINT0n<br>02h ADCTRIG2 - CPU1 Timer 1, TINT1n<br>03h ADCTRIG3 - CPU1 Timer 2, TINT2n<br>04h ADCTRIG4 - GPIO, ADCEXTSOC<br>05h ADCTRIG5 - ePWM1, ADCSOCA<br>06h ADCTRIG6 - ePWM1, ADCSOCB<br>07h ADCTRIG7 - ePWM2, ADCSOCA<br>08h ADCTRIG8 - ePWM2, ADCSOCB<br>09h ADCTRIG9 - ePWM3, ADCSOCA<br>0Ah ADCTRIG10 - ePWM3, ADCSOCB<br>0Bh ADCTRIG11 - ePWM4, ADCSOCA<br>0Ch ADCTRIG12 - ePWM4, ADCSOCB<br>0Dh ADCTRIG13 - ePWM5, ADCSOCA<br>0Eh ADCTRIG14 - ePWM5, ADCSOCB<br>0Fh ADCTRIG15 - ePWM6, ADCSOCA<br>10h ADCTRIG16 - ePWM6, ADCSOCB<br>11h ADCTRIG17 - ePWM7, ADCSOCA<br>12h ADCTRIG18 - ePWM7, ADCSOCB<br>13h ADCTRIG19 - ePWM8, ADCSOCA<br>14h ADCTRIG20 - ePWM8, ADCSOCB<br>15h ADCTRIG21 - ePWM9, ADCSOCA<br>16h ADCTRIG22 - ePWM9, ADCSOCB<br>17h ADCTRIG23 - ePWM10, ADCSOCA<br>18h ADCTRIG24 - ePWM10, ADCSOCB<br>19h ADCTRIG25 - ePWM11, ADCSOCA<br>1Ah ADCTRIG26 - ePWM11, ADCSOCB<br>1Bh ADCTRIG27 - ePWM12, ADCSOCA<br>1Ch ADCTRIG28 - ePWM12, ADCSOCB<br>1Dh ADCTRIG29 - CPU2 Timer 0, TINT0n<br>1Eh ADCTRIG30 - CPU2 Timer 1, TINT1n<br>1Fh ADCTRIG31 - CPU2 Timer 2, TINT2n<br>Reset type: SYSRSn |
| 19 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-69. ADCSOC14CTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 18-15 | CHSEL | R/W | 0h | SOC14 Channel Select. Selects the channel to be converted when SOC14 is received by the ADC.<br>Single-ended Signaling Mode (SIGNALMODE = 0):<br>0h ADCIN0<br>1h ADCIN1<br>2h ADCIN2<br>3h ADCIN3<br>4h ADCIN4<br>5h ADCIN5<br>6h ADCIN6<br>7h ADCIN7<br>8h ADCIN8<br>9h ADCIN9<br>Ah ADCIN10<br>Bh ADCIN11<br>Ch ADCIN12<br>Dh ADCIN13<br>Eh ADCIN14<br>Fh ADCIN15<br>Differential Signaling Mode (SIGNALMODE = 1):<br>0h ADCIN0 (non-inverting) and ADCIN1 (inverting)<br>1h ADCIN0 (non-inverting) and ADCIN1 (inverting)<br>2h ADCIN2 (non-inverting) and ADCIN3 (inverting)<br>3h ADCIN2 (non-inverting) and ADCIN3 (inverting)<br>4h ADCIN4 (non-inverting) and ADCIN5 (inverting)<br>5h ADCIN4 (non-inverting) and ADCIN5 (inverting)<br>6h ADCIN6 (non-inverting) and ADCIN7 (inverting)<br>7h ADCIN6 (non-inverting) and ADCIN7 (inverting)<br>8h ADCIN8 (non-inverting) and ADCIN9 (inverting)<br>9h ADCIN8 (non-inverting) and ADCIN9 (inverting)<br>Ah ADCIN10 (non-inverting) and ADCIN11 (inverting)<br>Bh ADCIN10 (non-inverting) and ADCIN11 (inverting)<br>Ch ADCIN12 (non-inverting) and ADCIN13 (inverting)<br>Dh ADCIN12 (non-inverting) and ADCIN13 (inverting)<br>Eh ADCIN14 (non-inverting) and ADCIN15 (inverting)<br>Fh ADCIN14 (non-inverting) and ADCIN15 (inverting)<br>Reset type: SYSRSn |
| 14-9 | RESERVED | R | 0h | Reserved |
| 8-0 | ACQPS | R/W | 0h | SOC14 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration.<br>000h Sample window is 1 system clock cycle wide<br>001h Sample window is 2 system clock cycles wide<br>002h Sample window is 3 system clock cycles wide<br>...<br>1FFh Sample window is 512 system clock cycles wide<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.32 ADCSOC15CTL Register (Offset = 2Eh) [Reset = 00000000h]

ADCSOC15CTL is shown in Figure 11-77 and described in Table 11-70.

Return to the Summary Table.

ADC SOC15 Control Register

**Figure 11-77. ADCSOC15CTL Register**

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

**Table 11-70. ADCSOC15CTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-25 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-70. ADCSOC15CTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 24-20 | TRIGSEL | R/W | 0h | SOC15 Trigger Source Select. Along with the SOC15 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC15 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.<br>Note: SOCFRC1 register can always be used to software trigger SOCs in addition to any hardware trigger configuration.<br>00h ADCTRIG0 - Software only<br>01h ADCTRIG1 - CPU1 Timer 0, TINT0n<br>02h ADCTRIG2 - CPU1 Timer 1, TINT1n<br>03h ADCTRIG3 - CPU1 Timer 2, TINT2n<br>04h ADCTRIG4 - GPIO, ADCEXTSOC<br>05h ADCTRIG5 - ePWM1, ADCSOCA<br>06h ADCTRIG6 - ePWM1, ADCSOCB<br>07h ADCTRIG7 - ePWM2, ADCSOCA<br>08h ADCTRIG8 - ePWM2, ADCSOCB<br>09h ADCTRIG9 - ePWM3, ADCSOCA<br>0Ah ADCTRIG10 - ePWM3, ADCSOCB<br>0Bh ADCTRIG11 - ePWM4, ADCSOCA<br>0Ch ADCTRIG12 - ePWM4, ADCSOCB<br>0Dh ADCTRIG13 - ePWM5, ADCSOCA<br>0Eh ADCTRIG14 - ePWM5, ADCSOCB<br>0Fh ADCTRIG15 - ePWM6, ADCSOCA<br>10h ADCTRIG16 - ePWM6, ADCSOCB<br>11h ADCTRIG17 - ePWM7, ADCSOCA<br>12h ADCTRIG18 - ePWM7, ADCSOCB<br>13h ADCTRIG19 - ePWM8, ADCSOCA<br>14h ADCTRIG20 - ePWM8, ADCSOCB<br>15h ADCTRIG21 - ePWM9, ADCSOCA<br>16h ADCTRIG22 - ePWM9, ADCSOCB<br>17h ADCTRIG23 - ePWM10, ADCSOCA<br>18h ADCTRIG24 - ePWM10, ADCSOCB<br>19h ADCTRIG25 - ePWM11, ADCSOCA<br>1Ah ADCTRIG26 - ePWM11, ADCSOCB<br>1Bh ADCTRIG27 - ePWM12, ADCSOCA<br>1Ch ADCTRIG28 - ePWM12, ADCSOCB<br>1Dh ADCTRIG29 - CPU2 Timer 0, TINT0n<br>1Eh ADCTRIG30 - CPU2 Timer 1, TINT1n<br>1Fh ADCTRIG31 - CPU2 Timer 2, TINT2n<br>Reset type: SYSRSn |
| 19 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-70. ADCSOC15CTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 18-15 | CHSEL | R/W | 0h | SOC15 Channel Select. Selects the channel to be converted when SOC15 is received by the ADC.<br>Single-ended Signaling Mode (SIGNALMODE = 0):<br>0h ADCIN0<br>1h ADCIN1<br>2h ADCIN2<br>3h ADCIN3<br>4h ADCIN4<br>5h ADCIN5<br>6h ADCIN6<br>7h ADCIN7<br>8h ADCIN8<br>9h ADCIN9<br>Ah ADCIN10<br>Bh ADCIN11<br>Ch ADCIN12<br>Dh ADCIN13<br>Eh ADCIN14<br>Fh ADCIN15<br>Differential Signaling Mode (SIGNALMODE = 1):<br>0h ADCIN0 (non-inverting) and ADCIN1 (inverting)<br>1h ADCIN0 (non-inverting) and ADCIN1 (inverting)<br>2h ADCIN2 (non-inverting) and ADCIN3 (inverting)<br>3h ADCIN2 (non-inverting) and ADCIN3 (inverting)<br>4h ADCIN4 (non-inverting) and ADCIN5 (inverting)<br>5h ADCIN4 (non-inverting) and ADCIN5 (inverting)<br>6h ADCIN6 (non-inverting) and ADCIN7 (inverting)<br>7h ADCIN6 (non-inverting) and ADCIN7 (inverting)<br>8h ADCIN8 (non-inverting) and ADCIN9 (inverting)<br>9h ADCIN8 (non-inverting) and ADCIN9 (inverting)<br>Ah ADCIN10 (non-inverting) and ADCIN11 (inverting)<br>Bh ADCIN10 (non-inverting) and ADCIN11 (inverting)<br>Ch ADCIN12 (non-inverting) and ADCIN13 (inverting)<br>Dh ADCIN12 (non-inverting) and ADCIN13 (inverting)<br>Eh ADCIN14 (non-inverting) and ADCIN15 (inverting)<br>Fh ADCIN14 (non-inverting) and ADCIN15 (inverting)<br>Reset type: SYSRSn |
| 14-9 | RESERVED | R | 0h | Reserved |
| 8-0 | ACQPS | R/W | 0h | SOC15 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration.<br>000h Sample window is 1 system clock cycle wide<br>001h Sample window is 2 system clock cycles wide<br>002h Sample window is 3 system clock cycles wide<br>...<br>1FFh Sample window is 512 system clock cycles wide<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.33 ADCEVTSTAT Register (Offset = 30h) [Reset = 0000h]

ADCEVTSTAT is shown in Figure 11-78 and described in Table 11-71.

Return to the Summary Table.

ADC Event Status Register

**Figure 11-78. 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 11-71. 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.<br>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.<br>Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority<br>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.<br>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.<br>Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority<br>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.<br>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.<br>Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority<br>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.<br>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.<br>Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-71. ADCEVTSTAT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 9 | PPB3TRIPLO | R | 0h | Post Processing Block 3 Trip Low Flag. When set indicates a digital compare trip low event has occurred.<br>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.<br>Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority<br>Reset type: SYSRSn |
| 8 | PPB3TRIPHI | R | 0h | Post Processing Block 3 Trip High Flag. When set indicates a digital compare trip high event has occurred.<br>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.<br>Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority<br>Reset type: SYSRSn |
| 7 | RESERVED | R | 0h | Reserved |
| 6 | PPB2ZERO | R | 0h | Post Processing Block 2 Zero Crossing Flag. When set indicates the ADCPPB2RESULT register has changed sign. This bit is gated by EOC signal.<br>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.<br>Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority<br>Reset type: SYSRSn |
| 5 | PPB2TRIPLO | R | 0h | Post Processing Block 2 Trip Low Flag. When set indicates a digital compare trip low event has occurred.<br>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.<br>Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority<br>Reset type: SYSRSn |
| 4 | PPB2TRIPHI | R | 0h | Post Processing Block 2 Trip High Flag. When set indicates a digital compare trip high event has occurred.<br>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.<br>Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority<br>Reset type: SYSRSn |
| 3 | RESERVED | R | 0h | Reserved |
| 2 | PPB1ZERO | R | 0h | Post Processing Block 1 Zero Crossing Flag. When set indicates the ADCPPB1RESULT register has changed sign. This bit is gated by EOC signal.<br>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.<br>Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-71. ADCEVTSTAT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1 | PPB1TRIPLO | R | 0h | Post Processing Block 1 Trip Low Flag. When set indicates a digital compare trip low event has occurred.<br>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.<br>Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority<br>Reset type: SYSRSn |
| 0 | PPB1TRIPHI | R | 0h | Post Processing Block 1 Trip High Flag. When set indicates a digital compare trip high event has occurred.<br>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.<br>Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.34 ADCEVTCLR Register (Offset = 32h) [Reset = 0000h]

ADCEVTCLR is shown in Figure 11-79 and described in Table 11-72.

Return to the Summary Table.

ADC Event Clear Register

**Figure 11-79. 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 11-72. 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.<br>Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority<br>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.<br>Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority<br>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.<br>Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority<br>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.<br>Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority<br>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.<br>Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority<br>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.<br>Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority<br>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.<br>Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-72. 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.<br>Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority<br>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.<br>Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority<br>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.<br>Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority<br>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.<br>Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority<br>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.<br>Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.35 ADCEVTSEL Register (Offset = 34h) [Reset = 0000h]

ADCEVTSEL is shown in Figure 11-80 and described in Table 11-73.

Return to the Summary Table.

ADC Event Selection Register

**Figure 11-80. 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 11-73. 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.<br>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.<br>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.<br>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.<br>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.<br>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.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-73. 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.<br>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.<br>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.<br>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.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.36 ADCEVTINTSEL Register (Offset = 36h) [Reset = 0000h]

ADCEVTINTSEL is shown in Figure 11-81 and described in Table 11-74.

Return to the Summary Table.

ADC Event Interrupt Selection Register

**Figure 11-81. 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 11-74. 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.<br>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.<br>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.<br>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.<br>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.<br>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.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-74. 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.<br>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.<br>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.<br>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.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.37 ADCOSDETECT Register (Offset = 38h) [Reset = 0000h]

ADCOSDETECT is shown in Figure 11-82 and described in Table 11-75.

Return to the Summary Table.

ADC Open and Shorts Detect Register

**Figure 11-82. 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 11-75. 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.<br>0h Open/Shorts detection circuit is disabled.<br>1h Open/Shorts detection circuit is enabled at zero scale.<br>2h Open/Shorts detection circuit is enabled at full scale.<br>3h Open/Shorts detection circuit is enabled at (nominal) 5/12 scale.<br>4h Open/Shorts detection circuit is enabled at (nominal) 7/12 scale.<br>5h Open/Shorts detection circuit is enabled with a (nominal) 5K pulldown to VSSA.<br>6h Open/Shorts detection circuit is enabled with a (nominal) 5K pullup to VDDA.<br>7h Open/Shorts detection circuit is enabled with a (nominal) 7K pulldown to VSSA.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.38 ADCCOUNTER Register (Offset = 39h) [Reset = 0000h]

ADCCOUNTER is shown in Figure 11-83 and described in Table 11-76.

Return to the Summary Table.

ADC Counter Register

**Figure 11-83. 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 11-76. 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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.39 ADCREV Register (Offset = 3Ah) [Reset = 0004h]

ADCREV is shown in Figure 11-84 and described in Table 11-77.

Return to the Summary Table.

ADC Revision Register

**Figure 11-84. ADCREV Register**

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

**Table 11-77. ADCREV Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | REV | R | 0h | ADC Revision. To allow documentation of differences between revisions. First version is labeled as 00h.<br>Reset type: SYSRSn |
| 7-0 | TYPE | R | 4h | ADC Type. Always set to 4 for this ADC.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.40 ADCOFFTRIM Register (Offset = 3Bh) [Reset = 0000h]

ADCOFFTRIM is shown in Figure 11-85 and described in Table 11-78.

Return to the Summary Table.

ADC Offset Trim Register

**Figure 11-85. 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 11-78. 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 | ADC Offset Trim. Adjusts the conversion results of the converter up or down to account for offset error in the ADC. A different offset trim is required for each combination of resolution and signal mode. Using the AdcSetMode function to set the resolution and signal mode will ensure that the correct offset trim is loaded.<br>Range is +127 steps to -128 steps (2's compliment format).<br>Regardless of the converter resolution, the size of each trim step is (VREFHI-VREFLO)/65536.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.41 ADCPPB1CONFIG Register (Offset = 40h) [Reset = 0000h]

ADCPPB1CONFIG is shown in Figure 11-86 and described in Table 11-79.

Return to the Summary Table.

ADC PPB1 Config Register

**Figure 11-86. ADCPPB1CONFIG Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | TWOSCOMPEN | CONFIG | | | |
| R-0h | | | R/W-0h | R/W-0h | | | |

**Table 11-79. ADCPPB1CONFIG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-5 | RESERVED | R | 0h | Reserved |
| 4 | TWOSCOMPEN | R/W | 0h | ADC Post Processing Block 1 Two's Complement Enable. When set this bit enables the post conversion hardware processing circuit that performs a two's complement on the output of the offset/reference subtraction unit before storing the result in the ADCPPB1RESULT register.<br>0 ADCPPB1RESULT = ADCRESULTx - ADCPPB1OFFREF<br>1 ADCPPB1RESULT = ADCPPB1OFFREF - ADCRESULTx<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-79. ADCPPB1CONFIG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-0 | CONFIG | R/W | 0h | ADC Post Processing Block 1 Configuration. This bit field defines which SOC/EOC/RESULT is assocatied with this post processing block.<br>0000 SOC0/EOC0/RESULT0 is associated with post processing block 1<br>0001 SOC1/EOC1/RESULT1 is associated with post processing block 1<br>0010 SOC2/EOC2/RESULT2 is associated with post processing block 1<br>0011 SOC3/EOC3/RESULT3 is associated with post processing block 1<br>0100 SOC4/EOC4/RESULT4 is associated with post processing block 1<br>0101 SOC5/EOC5/RESULT5 is associated with post processing block 1<br>0110 SOC6/EOC6/RESULT6 is associated with post processing block 1<br>0111 SOC7/EOC7/RESULT7 is associated with post processing block 1<br>1000 SOC8/EOC8/RESULT8 is associated with post processing block 1<br>1001 SOC9/EOC9/RESULT9 is associated with post processing block 1<br>1010 SOC10/EOC10/RESULT10 is associated with post processing block 1<br>1011 SOC11/EOC11/RESULT11 is associated with post processing block 1<br>1100 SOC12/EOC12/RESULT12 is associated with post processing block 1<br>1101 SOC13/EOC13/RESULT13 is associated with post processing block 1<br>1110 SOC14/EOC14/RESULT14 is associated with post processing block 1<br>1111 SOC15/EOC15/RESULT15 is associated with post processing block 1<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.42 ADCPPB1STAMP Register (Offset = 41h) [Reset = 0000h]

ADCPPB1STAMP is shown in Figure 11-87 and described in Table 11-80.

Return to the Summary Table.

ADC PPB1 Sample Delay Time Stamp Register

**Figure 11-87. 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 11-80. 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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.43 ADCPPB1OFFCAL Register (Offset = 42h) [Reset = 0000h]

ADCPPB1OFFCAL is shown in Figure 11-88 and described in Table 11-81.

Return to the Summary Table.

ADC PPB1 Offset Calibration Register

**Figure 11-88. 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 11-81. ADCPPB1OFFCAL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-10 | RESERVED | R | 0h | Reserved |
| 9-0 | OFFCAL | R/W | 0h | 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.<br>000h No change. The ADC output is stored directly into ADCRESULT.<br>001h ADC output - 1 is stored into ADCRESULT.<br>002h ADC output - 2 is stored into ADCRESULT.<br>...<br>200h ADC output + 512 is stored into ADCRESULT.<br>...<br>3FFh ADC output + 1 is stored into ADCRESULT.<br>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.<br>Note: in the case that multiple PPBs point to the same SOC, only the OFFCAL of the highest numbered PPB will be applied.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.44 ADCPPB1OFFREF Register (Offset = 43h) [Reset = 0000h]

ADCPPB1OFFREF is shown in Figure 11-89 and described in Table 11-82.

Return to the Summary Table.

ADC PPB1 Offset Reference Register

**Figure 11-89. 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 11-82. 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.<br>0000h No change. The ADCRESULT value is passed on.<br>0001h ADCRESULT - 1 is passed on.<br>0002h ADCRESULT - 2 is passed on.<br>...<br>8000h ADCRESULT - 32,768 is passed on.<br>...<br>FFFFh ADCRESULT - 65,535 is passed on.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.45 ADCPPB1TRIPHI Register (Offset = 44h) [Reset = 00000000h]

ADCPPB1TRIPHI is shown in Figure 11-90 and described in Table 11-83.

Return to the Summary Table.

ADC PPB1 Trip High Register

**Figure 11-90. 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 11-83. 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.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.46 ADCPPB1TRIPLO Register (Offset = 46h) [Reset = 00000000h]

ADCPPB1TRIPLO is shown in Figure 11-91 and described in Table 11-84.

Return to the Summary Table.

ADC PPB1 Trip Low/Trigger Time Stamp Register

**Figure 11-91. 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 11-84. 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.<br>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.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.47 ADCPPB2CONFIG Register (Offset = 48h) [Reset = 0000h]

ADCPPB2CONFIG is shown in Figure 11-92 and described in Table 11-85.

Return to the Summary Table.

ADC PPB2 Config Register

**Figure 11-92. ADCPPB2CONFIG Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | TWOSCOMPEN | CONFIG | | | |
| R-0h | | | R/W-0h | R/W-0h | | | |

**Table 11-85. ADCPPB2CONFIG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-5 | RESERVED | R | 0h | Reserved |
| 4 | TWOSCOMPEN | R/W | 0h | ADC Post Processing Block 2 Two's Complement Enable. When set this bit enables the post conversion hardware processing circuit that performs a two's complement on the output of the offset/reference subtraction unit before storing the result in the ADCPPB2RESULT register.<br>0 ADCPPB2RESULT = ADCRESULTx - ADCPPB2OFFREF<br>1 ADCPPB2RESULT = ADCPPB2OFFREF - ADCRESULTx<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-85. 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 assocatied with this post processing block.<br>0000 SOC0/EOC0/RESULT0 is associated with post processing block 2<br>0001 SOC1/EOC1/RESULT1 is associated with post processing block 2<br>0010 SOC2/EOC2/RESULT2 is associated with post processing block 2<br>0011 SOC3/EOC3/RESULT3 is associated with post processing block 2<br>0100 SOC4/EOC4/RESULT4 is associated with post processing block 2<br>0101 SOC5/EOC5/RESULT5 is associated with post processing block 2<br>0110 SOC6/EOC6/RESULT6 is associated with post processing block 2<br>0111 SOC7/EOC7/RESULT7 is associated with post processing block 2<br>1000 SOC8/EOC8/RESULT8 is associated with post processing block 2<br>1001 SOC9/EOC9/RESULT9 is associated with post processing block 2<br>1010 SOC10/EOC10/RESULT10 is associated with post processing block 2<br>1011 SOC11/EOC11/RESULT11 is associated with post processing block 2<br>1100 SOC12/EOC12/RESULT12 is associated with post processing block 2<br>1101 SOC13/EOC13/RESULT13 is associated with post processing block 2<br>1110 SOC14/EOC14/RESULT14 is associated with post processing block 2<br>1111 SOC15/EOC15/RESULT15 is associated with post processing block 2<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.48 ADCPPB2STAMP Register (Offset = 49h) [Reset = 0000h]

ADCPPB2STAMP is shown in Figure 11-93 and described in Table 11-86.

Return to the Summary Table.

ADC PPB2 Sample Delay Time Stamp Register

**Figure 11-93. 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 11-86. 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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.49 ADCPPB2OFFCAL Register (Offset = 4Ah) [Reset = 0000h]

ADCPPB2OFFCAL is shown in Figure 11-94 and described in Table 11-87.

Return to the Summary Table.

ADC PPB2 Offset Calibration Register

**Figure 11-94. 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 11-87. 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.<br>000h No change. The ADC output is stored directly into ADCRESULT.<br>001h ADC output - 1 is stored into ADCRESULT.<br>002h ADC output - 2 is stored into ADCRESULT.<br>...<br>200h ADC output + 512 is stored into ADCRESULT.<br>...<br>3FFh ADC output + 1 is stored into ADCRESULT.<br>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.<br>Note: in the case that multiple PPBs point to the same SOC, only the OFFCAL of the highest numbered PPB will be applied.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.50 ADCPPB2OFFREF Register (Offset = 4Bh) [Reset = 0000h]

ADCPPB2OFFREF is shown in Figure 11-95 and described in Table 11-88.

Return to the Summary Table.

ADC PPB2 Offset Reference Register

**Figure 11-95. 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 11-88. ADCPPB2OFFREF Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | OFFREF | R/W | 0h | 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.<br>0000h No change. The ADCRESULT value is passed on.<br>0001h ADCRESULT - 1 is passed on.<br>0002h ADCRESULT - 2 is passed on.<br>...<br>8000h ADCRESULT - 32,768 is passed on.<br>...<br>FFFFh ADCRESULT - 65,535 is passed on.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.51 ADCPPB2TRIPHI Register (Offset = 4Ch) [Reset = 00000000h]

ADCPPB2TRIPHI is shown in Figure 11-96 and described in Table 11-89.

Return to the Summary Table.

ADC PPB2 Trip High Register

**Figure 11-96. 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 11-89. 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.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.52 ADCPPB2TRIPLO Register (Offset = 4Eh) [Reset = 00000000h]

ADCPPB2TRIPLO is shown in Figure 11-97 and described in Table 11-90.

Return to the Summary Table.

ADC PPB2 Trip Low/Trigger Time Stamp Register

**Figure 11-97. 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 11-90. 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.<br>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.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.53 ADCPPB3CONFIG Register (Offset = 50h) [Reset = 0000h]

ADCPPB3CONFIG is shown in Figure 11-98 and described in Table 11-91.

Return to the Summary Table.

ADC PPB3 Config Register

**Figure 11-98. ADCPPB3CONFIG Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | TWOSCOMPEN | CONFIG | | | |
| R-0h | | | R/W-0h | R/W-0h | | | |

**Table 11-91. ADCPPB3CONFIG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-5 | RESERVED | R | 0h | Reserved |
| 4 | TWOSCOMPEN | R/W | 0h | ADC Post Processing Block 3 Two's Complement Enable. When set this bit enables the post conversion hardware processing circuit that performs a two's complement on the output of the offset/reference subtraction unit before storing the result in the ADCPPB3RESULT register.<br>0 ADCPPB3RESULT = ADCRESULTx - ADCPPB3OFFREF<br>1 ADCPPB3RESULT = ADCPPB3OFFREF - ADCRESULTx<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-91. ADCPPB3CONFIG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-0 | CONFIG | R/W | 0h | ADC Post Processing Block 3 Configuration. This bit field defines which SOC/EOC/RESULT is assocatied with this post processing block.<br>0000 SOC0/EOC0/RESULT0 is associated with post processing block 3<br>0001 SOC1/EOC1/RESULT1 is associated with post processing block 3<br>0010 SOC2/EOC2/RESULT2 is associated with post processing block 3<br>0011 SOC3/EOC3/RESULT3 is associated with post processing block 3<br>0100 SOC4/EOC4/RESULT4 is associated with post processing block 3<br>0101 SOC5/EOC5/RESULT5 is associated with post processing block 3<br>0110 SOC6/EOC6/RESULT6 is associated with post processing block 3<br>0111 SOC7/EOC7/RESULT7 is associated with post processing block 3<br>1000 SOC8/EOC8/RESULT8 is associated with post processing block 3<br>1001 SOC9/EOC9/RESULT9 is associated with post processing block 3<br>1010 SOC10/EOC10/RESULT10 is associated with post processing block 3<br>1011 SOC11/EOC11/RESULT11 is associated with post processing block 3<br>1100 SOC12/EOC12/RESULT12 is associated with post processing block 3<br>1101 SOC13/EOC13/RESULT13 is associated with post processing block 3<br>1110 SOC14/EOC14/RESULT14 is associated with post processing block 3<br>1111 SOC15/EOC15/RESULT15 is associated with post processing block 3<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.54 ADCPPB3STAMP Register (Offset = 51h) [Reset = 0000h]

ADCPPB3STAMP is shown in Figure 11-99 and described in Table 11-92.

Return to the Summary Table.

ADC PPB3 Sample Delay Time Stamp Register

**Figure 11-99. 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 11-92. 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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.55 ADCPPB3OFFCAL Register (Offset = 52h) [Reset = 0000h]

ADCPPB3OFFCAL is shown in Figure 11-100 and described in Table 11-93.

Return to the Summary Table.

ADC PPB3 Offset Calibration Register

**Figure 11-100. 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 11-93. ADCPPB3OFFCAL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-10 | RESERVED | R | 0h | Reserved |
| 9-0 | OFFCAL | R/W | 0h | 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.<br>000h No change. The ADC output is stored directly into ADCRESULT.<br>001h ADC output - 1 is stored into ADCRESULT.<br>002h ADC output - 2 is stored into ADCRESULT.<br>...<br>200h ADC output + 512 is stored into ADCRESULT.<br>...<br>3FFh ADC output + 1 is stored into ADCRESULT.<br>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.<br>Note: in the case that multiple PPBs point to the same SOC, only the OFFCAL of the highest numbered PPB will be applied.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.56 ADCPPB3OFFREF Register (Offset = 53h) [Reset = 0000h]

ADCPPB3OFFREF is shown in Figure 11-101 and described in Table 11-94.

Return to the Summary Table.

ADC PPB3 Offset Reference Register

**Figure 11-101. 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 11-94. 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.<br>0000h No change. The ADCRESULT value is passed on.<br>0001h ADCRESULT - 1 is passed on.<br>0002h ADCRESULT - 2 is passed on.<br>...<br>8000h ADCRESULT - 32,768 is passed on.<br>...<br>FFFFh ADCRESULT - 65,535 is passed on.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.57 ADCPPB3TRIPHI Register (Offset = 54h) [Reset = 00000000h]

ADCPPB3TRIPHI is shown in Figure 11-102 and described in Table 11-95.

Return to the Summary Table.

ADC PPB3 Trip High Register

**Figure 11-102. 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 11-95. 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.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.58 ADCPPB3TRIPLO Register (Offset = 56h) [Reset = 00000000h]

ADCPPB3TRIPLO is shown in Figure 11-103 and described in Table 11-96.

Return to the Summary Table.

ADC PPB3 Trip Low/Trigger Time Stamp Register

**Figure 11-103. 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 11-96. 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.<br>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.<br>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 PPBRSULT bit field of the ADCPPB3RESULT register.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.59 ADCPPB4CONFIG Register (Offset = 58h) [Reset = 0000h]

ADCPPB4CONFIG is shown in Figure 11-104 and described in Table 11-97.

Return to the Summary Table.

ADC PPB4 Config Register

**Figure 11-104. ADCPPB4CONFIG Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | TWOSCOMPEN | CONFIG | | | |
| R-0h | | | R/W-0h | R/W-0h | | | |

**Table 11-97. ADCPPB4CONFIG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-5 | RESERVED | R | 0h | Reserved |
| 4 | TWOSCOMPEN | R/W | 0h | ADC Post Processing Block 4 Two's Complement Enable. When set this bit enables the post conversion hardware processing circuit that performs a two's complement on the output of the offset/reference subtraction unit before storing the result in the ADCPPB4RESULT register.<br>0 ADCPPB4RESULT = ADCRESULTx - ADCPPB4OFFREF<br>1 ADCPPB4RESULT = ADCPPB4OFFREF - ADCRESULTx<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-97. 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 assocatied with this post processing block.<br>0000 SOC0/EOC0/RESULT0 is associated with post processing block 4<br>0001 SOC1/EOC1/RESULT1 is associated with post processing block 4<br>0010 SOC2/EOC2/RESULT2 is associated with post processing block 4<br>0011 SOC3/EOC3/RESULT3 is associated with post processing block 4<br>0100 SOC4/EOC4/RESULT4 is associated with post processing block 4<br>0101 SOC5/EOC5/RESULT5 is associated with post processing block 4<br>0110 SOC6/EOC6/RESULT6 is associated with post processing block 4<br>0111 SOC7/EOC7/RESULT7 is associated with post processing block 4<br>1000 SOC8/EOC8/RESULT8 is associated with post processing block 4<br>1001 SOC9/EOC9/RESULT9 is associated with post processing block 4<br>1010 SOC10/EOC10/RESULT10 is associated with post processing block 4<br>1011 SOC11/EOC11/RESULT11 is associated with post processing block 4<br>1100 SOC12/EOC12/RESULT12 is associated with post processing block 4<br>1101 SOC13/EOC13/RESULT13 is associated with post processing block 4<br>1110 SOC14/EOC14/RESULT14 is associated with post processing block 4<br>1111 SOC15/EOC15/RESULT15 is associated with post processing block 4<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.60 ADCPPB4STAMP Register (Offset = 59h) [Reset = 0000h]

ADCPPB4STAMP is shown in Figure 11-105 and described in Table 11-98.

Return to the Summary Table.

ADC PPB4 Sample Delay Time Stamp Register

**Figure 11-105. 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 11-98. 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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.61 ADCPPB4OFFCAL Register (Offset = 5Ah) [Reset = 0000h]

ADCPPB4OFFCAL is shown in Figure 11-106 and described in Table 11-99.

Return to the Summary Table.

ADC PPB4 Offset Calibration Register

**Figure 11-106. 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 11-99. 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.<br>000h No change. The ADC output is stored directly into ADCRESULT.<br>001h ADC output - 1 is stored into ADCRESULT.<br>002h ADC output - 2 is stored into ADCRESULT.<br>...<br>200h ADC output + 512 is stored into ADCRESULT.<br>...<br>3FFh ADC output + 1 is stored into ADCRESULT.<br>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.<br>Note: in the case that multiple PPBs point to the same SOC, only the OFFCAL of the highest numbered PPB will be applied.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.62 ADCPPB4OFFREF Register (Offset = 5Bh) [Reset = 0000h]

ADCPPB4OFFREF is shown in Figure 11-107 and described in Table 11-100.

Return to the Summary Table.

ADC PPB4 Offset Reference Register

**Figure 11-107. 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 11-100. ADCPPB4OFFREF Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | OFFREF | R/W | 0h | 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.<br>0000h No change. The ADCRESULT value is passed on.<br>0001h ADCRESULT - 1 is passed on.<br>0002h ADCRESULT - 2 is passed on.<br>...<br>8000h ADCRESULT - 32,768 is passed on.<br>...<br>FFFFh ADCRESULT - 65,535 is passed on.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.63 ADCPPB4TRIPHI Register (Offset = 5Ch) [Reset = 00000000h]

ADCPPB4TRIPHI is shown in Figure 11-108 and described in Table 11-101.

Return to the Summary Table.

ADC PPB4 Trip High Register

**Figure 11-108. 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 11-101. 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.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.64 ADCPPB4TRIPLO Register (Offset = 5Eh) [Reset = 00000000h]

ADCPPB4TRIPLO is shown in Figure 11-109 and described in Table 11-102.

Return to the Summary Table.

ADC PPB4 Trip Low/Trigger Time Stamp Register

**Figure 11-109. 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 11-102. 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.<br>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.<br>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 PPBRSULT bit field of the ADCPPB4RESULT register.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.65 ADCINLTRIM1 Register (Offset = 70h) [Reset = 00000000h]

ADCINLTRIM1 is shown in Figure 11-110 and described in Table 11-103.

Return to the Summary Table.

ADC Linearity Trim 1 Register

**Figure 11-110. ADCINLTRIM1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| INLTRIM31TO0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 11-103. ADCINLTRIM1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | INLTRIM31TO0 | R/W | 0h | ADC Linearity Trim Bits 31-0.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.66 ADCINLTRIM2 Register (Offset = 72h) [Reset = 00000000h]

ADCINLTRIM2 is shown in Figure 11-111 and described in Table 11-104.

Return to the Summary Table.

ADC Linearity Trim 2 Register

**Figure 11-111. 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 11-104. ADCINLTRIM2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | INLTRIM63TO32 | R/W | 0h | ADC Linearity Trim Bits 63-32.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.67 ADCINLTRIM3 Register (Offset = 74h) [Reset = 00000000h]

ADCINLTRIM3 is shown in Figure 11-112 and described in Table 11-105.

Return to the Summary Table.

ADC Linearity Trim 3 Register

**Figure 11-112. 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 11-105. ADCINLTRIM3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | INLTRIM95TO64 | R/W | 0h | ADC Linearity Trim Bits 95-64.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.68 ADCINLTRIM4 Register (Offset = 76h) [Reset = 00000000h]

ADCINLTRIM4 is shown in Figure 11-113 and described in Table 11-106.

Return to the Summary Table.

ADC Linearity Trim 4 Register

**Figure 11-113. ADCINLTRIM4 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| INLTRIM127TO96 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 11-106. ADCINLTRIM4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | INLTRIM127TO96 | R/W | 0h | ADC Linearity Trim Bits 127-96.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.69 ADCINLTRIM5 Register (Offset = 78h) [Reset = 00000000h]

ADCINLTRIM5 is shown in Figure 11-114 and described in Table 11-107.

Return to the Summary Table.

ADC Linearity Trim 5 Register

**Figure 11-114. ADCINLTRIM5 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| INLTRIM159TO128 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 11-107. ADCINLTRIM5 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | INLTRIM159TO128 | R/W | 0h | ADC Linearity Trim Bits 159-128.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 11.15.3.70 ADCINLTRIM6 Register (Offset = 7Ah) [Reset = 00000000h]

ADCINLTRIM6 is shown in Figure 11-115 and described in Table 11-108.

Return to the Summary Table.

ADC Linearity Trim 6 Register

**Figure 11-115. ADCINLTRIM6 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| INLTRIM191TO160 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 11-108. ADCINLTRIM6 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | INLTRIM191TO160 | R/W | 0h | ADC Linearity Trim Bits 191-160.<br>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.<br>Reset type: SYSRSn |

### *11.15.4 ADC Registers to Driverlib Functions*

**Table 11-109. ADC Registers to Driverlib Functions**

| File | Driverlib Function |
|---|---|
| **ADCCTL1** | |
| adc.h | ADC_setInterruptPulseMode |
| adc.h | ADC_enableConverter |
| adc.h | ADC_disableConverter |
| adc.h | ADC_isBusy |
| **ADCCTL2** | |
| adc.c | ADC_setMode |
| adc.c | ADC_setINLTrim |
| adc.c | ADC_setOffsetTrim |
| adc.h | ADC_setPrescaler |
| **ADCBURSTCTL** | |
| adc.h | ADC_setBurstModeConfig |
| adc.h | ADC_enableBurstMode |
| adc.h | ADC_disableBurstMode |
| **ADCINTFLG** | |
| adc.h | ADC_getInterruptStatus |
| adc.h | ADC_clearInterruptStatus |
| **ADCINTFLGCLR** | |
| adc.h | ADC_clearInterruptStatus |
| **ADCINTOVF** | |
| adc.h | ADC_getInterruptOverflowStatus |
| adc.h | ADC_clearInterruptOverflowStatus |
| **ADCINTOVFCLR** | |
| adc.h | ADC_clearInterruptOverflowStatus |
| **ADCINTSEL1N2** | |
| adc.h | ADC_enableInterrupt |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-109. ADC Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| adc.h | ADC_disableInterrupt |
| adc.h | ADC_setInterruptSource |
| adc.h | ADC_enableContinuousMode |
| adc.h | ADC_disableContinuousMode |
| **ADCINTSEL3N4** | |
| - | See INTSEL1N2 |
| **ADCSOCPRICTL** | |
| adc.h | ADC_setSOCPriority |
| **ADCINTSOCSEL1** | |
| adc.h | ADC_setInterruptSOCTrigger |
| **ADCINTSOCSEL2** | |
| - | See INTSOCSEL1 |
| **ADCSOCFLG1** | |
| - | |
| **ADCSOCFRC1** | |
| adc.h | ADC_forceSOC |
| adc.h | ADC_forceMultipleSOC |
| **ADCSOCOVF1** | |
| - | |
| **ADCSOCOVFCLR1** | |
| - | |
| **ADCSOC0CTL** | |
| adc.h | ADC_setupSOC |
| **ADCSOC1CTL** | |
| - | See SOC0CTL |
| **ADCSOC2CTL** | |
| - | See SOC0CTL |
| **ADCSOC3CTL** | |
| - | See SOC0CTL |
| **ADCSOC4CTL** | |
| - | See SOC0CTL |
| **ADCSOC5CTL** | |
| - | See SOC0CTL |
| **ADCSOC6CTL** | |
| - | See SOC0CTL |
| **ADCSOC7CTL** | |
| - | See SOC0CTL |
| **ADCSOC8CTL** | |
| - | See SOC0CTL |
| **ADCSOC9CTL** | |
| - | See SOC0CTL |
| **ADCSOC10CTL** | |
| - | See SOC0CTL |
| **ADCSOC11CTL** | |
| - | See SOC0CTL |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-109. ADC Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| **ADCSOC12CTL** | |
| - | See SOC0CTL |
| **ADCSOC13CTL** | |
| - | See SOC0CTL |
| **ADCSOC14CTL** | |
| - | See SOC0CTL |
| **ADCSOC15CTL** | |
| - | See SOC0CTL |
| **ADCEVTSTAT** | |
| adc.h | ADC_getPPBEventStatus |
| **ADCEVTCLR** | |
| adc.h | ADC_clearPPBEventStatus |
| **ADCEVTSEL** | |
| adc.h | ADC_enablePPBEvent |
| adc.h | ADC_disablePPBEvent |
| **ADCEVTINTSEL** | |
| adc.h | ADC_enablePPBEventInterrupt |
| adc.h | ADC_disablePPBEventInterrupt |
| **ADCOSDETECT** | |
| adc.h | ADC_configOSDetectMode |
| **ADCCOUNTER** | |
| - | |
| **ADCREV** | |
| - | |
| **ADCOFFTRIM** | |
| adc.c | ADC_setOffsetTrim |
| **ADCPPB1CONFIG** | |
| adc.h | ADC_setupPPB |
| adc.h | ADC_enablePPBTwosComplement |
| adc.h | ADC_disablePPBTwosComplement |
| **ADCPPB1STAMP** | |
| adc.h | ADC_getPPBDelayTimeStamp |
| **ADCPPB1OFFCAL** | |
| adc.h | ADC_setPPBCalibrationOffset |
| **ADCPPB1OFFREF** | |
| adc.h | ADC_setPPBReferenceOffset |
| **ADCPPB1TRIPHI** | |
| adc.c | ADC_setPPBTripLimits |
| **ADCPPB1TRIPLO** | |
| adc.c | ADC_setPPBTripLimits |
| **ADCPPB2CONFIG** | |
| - | See PPB1CONFIG |
| **ADCPPB2STAMP** | |
| - | See PPB1STAMP |
| **ADCPPB2OFFCAL** | |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-109. ADC Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| - | See PPB1OFFCAL |
| **ADCPPB2OFFREF** | |
| - | See PPB1OFFREF |
| **ADCPPB2TRIPHI** | |
| - | See PPB1TRIPHI |
| **ADCPPB2TRIPLO** | |
| - | See PPB1TRIPLO |
| **ADCPPB3CONFIG** | |
| - | See PPB1CONFIG |
| **ADCPPB3STAMP** | |
| - | See PPB1STAMP |
| **ADCPPB3OFFCAL** | |
| - | See PPB1OFFCAL |
| **ADCPPB3OFFREF** | |
| - | See PPB1OFFREF |
| **ADCPPB3TRIPHI** | |
| - | See PPB1TRIPHI |
| **ADCPPB3TRIPLO** | |
| - | See PPB1TRIPLO |
| **ADCPPB4CONFIG** | |
| - | See PPB1CONFIG |
| **ADCPPB4STAMP** | |
| - | See PPB1STAMP |
| **ADCPPB4OFFCAL** | |
| - | See PPB1OFFCAL |
| **ADCPPB4OFFREF** | |
| - | See PPB1OFFREF |
| **ADCPPB4TRIPHI** | |
| - | See PPB1TRIPHI |
| **ADCPPB4TRIPLO** | |
| - | See PPB1TRIPLO |
| **ADCINLTRIM1** | |
| adc.c | ADC_setINLTrim |
| **ADCINLTRIM2** | |
| adc.c | ADC_setINLTrim |
| **ADCINLTRIM3** | |
| - | |
| **ADCINLTRIM4** | |
| adc.c | ADC_setINLTrim |
| **ADCINLTRIM5** | |
| adc.c | ADC_setINLTrim |
| **ADCINLTRIM6** | |
| - | |
| **ADCRESULT0** | |
| adc.h | ADC_readResult |



Copyright © 2024 Texas Instruments Incorporated

**Table 11-109. ADC Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| **ADCRESULT1** | |
| - | See RESULT0 |
| **ADCRESULT2** | |
| - | See RESULT0 |
| **ADCRESULT3** | |
| - | See RESULT0 |
| **ADCRESULT4** | |
| - | See RESULT0 |
| **ADCRESULT5** | |
| - | See RESULT0 |
| **ADCRESULT6** | |
| - | See RESULT0 |
| **ADCRESULT7** | |
| - | See RESULT0 |
| **ADCRESULT8** | |
| - | See RESULT0 |
| **ADCRESULT9** | |
| - | See RESULT0 |
| **ADCRESULT10** | |
| - | See RESULT0 |
| **ADCRESULT11** | |
| - | See RESULT0 |
| **ADCRESULT12** | |
| - | See RESULT0 |
| **ADCRESULT13** | |
| - | See RESULT0 |
| **ADCRESULT14** | |
| - | See RESULT0 |
| **ADCRESULT15** | |
| - | See RESULT0 |
| **ADCPPB1RESULT** | |
| adc.h | ADC_readPPBResult |
| **ADCPPB2RESULT** | |
| - | See PPB1RESULT |
| **ADCPPB3RESULT** | |
| - | See PPB1RESULT |
| **ADCPPB4RESULT** | |
| - | See PPB1RESULT |



Copyright © 2024 Texas Instruments Incorporated

Chapter 12

# Buffered Digital-to-Analog Converter (DAC)

The buffered digital-to-analog converter (DAC) is an analog module that can output a programmable, arbitrary reference voltage.


**12.1 Introduction**....1790
**12.2 Using the DAC**....1791
**12.3 Lock Registers**....1792
**12.4 Software**....1793
**12.5 DAC Registers**....1793





Copyright © 2024 Texas Instruments Incorporated

## 12.1 Introduction

The buffered DAC module consists of an internal 12-bit DAC and an analog output buffer that is capable of driving an external load. An integrated pull-down resistor on the DAC output helps to provide a known pin voltage when the output buffer is disabled. This pull-down resistor cannot be disabled and remains as a passive component on the pin, even for other shared pinmux functions. The buffered DAC is a general-purpose DAC that can be used to generate a DC voltage in addition to AC waveforms such as sine waves, square waves, triangle waves and so forth. Software writes to the DAC value register can take effect immediately or can be synchronized with EPWMSYNCPER events.

### *12.1.1 DAC Related Collateral*

**Foundational Materials**

- C2000 Academy - DAC
- *High Speed, Digital to Analog Converters Basics Application Report*
- *Real-Time Control Reference Guide*
  - Refer to the DAC section
- *Understanding Data Converters Application Report*

**Getting Started Materials**

- MathWorks F2807x/F2837xD/F2837xS/F28004x/F2838x DAC
  - NOTE: This is a non-TI (third party) site.

### *12.1.2 Features*

Each buffered DAC has the following features:

- 12-bit programmable internal DAC
- Selectable reference voltage source
- Pull-down resistor on output
- Ability to synchronize with EPWMSYNCPER

### *12.1.3 Block Diagram*

The block diagram for the buffered DAC is shown in Figure 12-1.



Copyright © 2024 Texas Instruments Incorporated

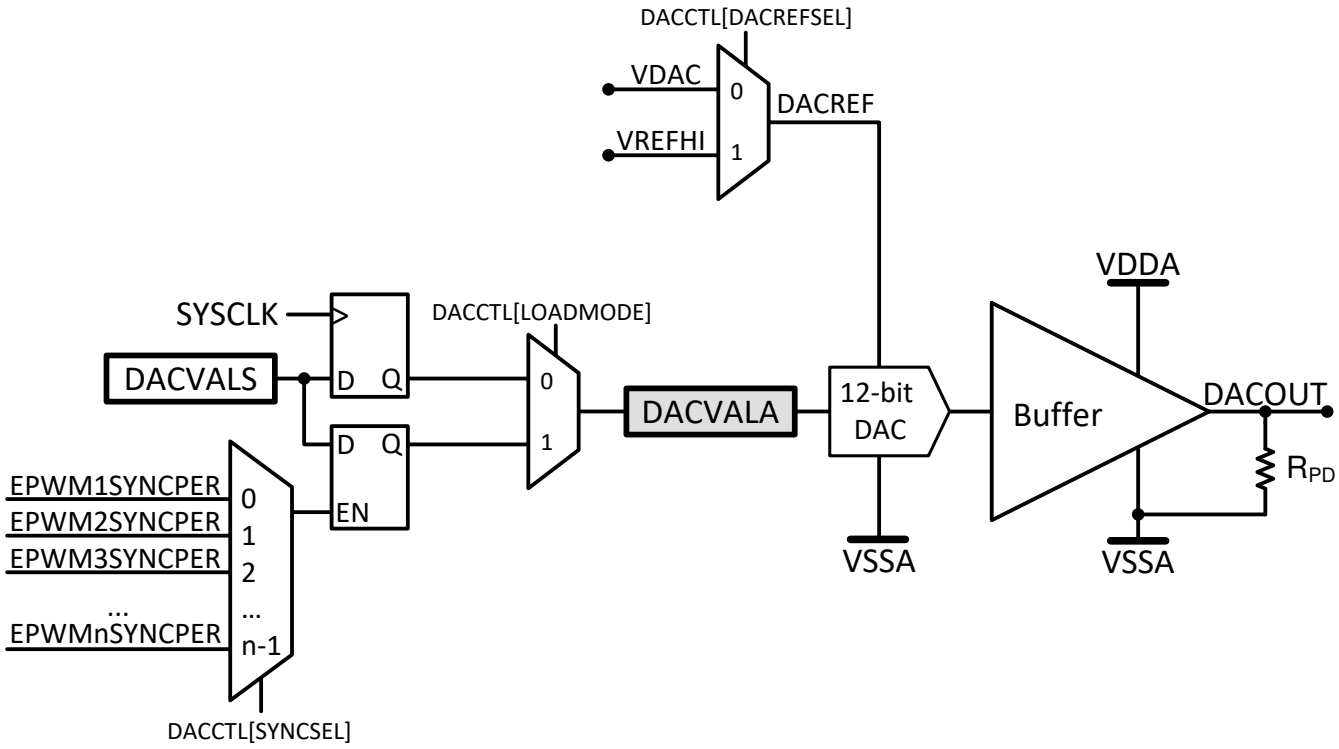



**Figure 12-1. DAC Module Block Diagram**

## 12.2 Using the DAC

The internal DAC's reference voltage source, DACREF, is selectable between VDAC and VREFHI.

Two sets of DACVAL registers, DACVALA and DACVALS, are present in the buffered DAC module. DACVALA is a read-only register that actively controls the buffered DAC value. DACVALS is a writable shadow register that loads into DACVALA either immediately or synchronized with the next EPWMSYNCPER event. If the clock to the buffered DAC is disabled while the buffered DAC is outputting a voltage, the output voltage remains unaffected, but DACVALA and DACVALS is no longer updated with register writes. Enabling the clock to the buffered DAC restores the DAC to the state before the clock was disabled.

The output of the internal DAC is calculated with the following equation:

$$DACOUT = \frac{DACVALA * DACREF}{4096} \quad (15)$$

The output buffer of the buffered DAC can exhibit non-linear behavior near the supply rails (VDDA/VSSA). To determine the linear range of the buffered DAC, see the device-specific data sheet.

### *12.2.1 Initialization Sequence*

1. Enable the buffered DAC clock.
2. Set DACREF with DACREFSEL.
3. Power up the buffered DAC with DACOUTEN.
4. Wait for the power-up time to elapse before outputting a voltage. To determine the power-up time of the buffered DAC, see the device data sheet.
5. For predictable behavior of the buffered DAC, consecutive writes to DACVALS must be spaced apart according to the settling time of the buffered DAC. To determine the settling time of the buffered DAC, see the device data sheet.



Copyright © 2024 Texas Instruments Incorporated

### 12.2.2 DAC Offset Adjustment

Zero offset error is defined as the difference between the voltage at midcode (2048) and 1.25V (for 2.5V reference voltage). DAC offset error is calibrated at 2.5V reference voltage and loaded into the DAC offset trim register as part of the Device_cal() function. If the DAC is used at any reference voltage other than 2.5V, the offset trim must be adjusted to make sure the offset error performance stays within the device-specific data sheet limits. The DAC offset register is a 16-bit register that contains the 8-bit signed offset trim in the lower half of the register. Use the function call DAC_tuneOffsetTrim() found in C2000Ware to adjust the offset.

### 12.2.3 EPWMSYNCPER Signal

EPWMSYNCPER comes from the Time-Base submodule of the EPWM. For a detailed description of how this signal is generated, refer to the *Time-Base Submodule* section of the *Enhanced Pulse Width Modulator (ePWM)* chapter.

The EPWMSYNCPER signal that loads DACVALA when DACCTL [LOADMODE] = 1 is a level trigger load. If TBCTR and TBPRD of the EPWM are both 0, EPWMSYNCPER is held at a high level and DACVALA is immediately loaded from DACVALS irrespective of the value of DACCTL [LOADMODE]. Due to this, configure the EPWM first before setting DACCTL [LOADMODE] to 1.

**Note**

The name of the sync signal that the GPDAC receives from the EPWM has been updated from PWMSYNC to EPWMSYNCPER (SYNCPER/PWMSYNCPER/EPWMxSYNCPER) to avoid confusion with the other EPWM sync signals EPWMSYNCI and EPWMSYNCO. For a description of these signals, see the *Enhanced Pulse Width Modulator (ePWM)* chapter.

## 12.3 Lock Registers

A DACLOCK register is provided to prevent spurious writes from modifying the DACCTL, DACVALS, and DACOUTEN registers. Once a register is protected through DACLOCK, write access are locked out until the device is reset.



Copyright © 2024 Texas Instruments Incorporated

## 12.4 Software

### *12.4.1 DAC Examples*

NOTE: These examples are located in the C2000Ware installation at the following location: C2000Ware_VERSION#/driverlib/DEVICE_GPN/examples/CORE_IF_MULTICORE/dac

Cloud access to these examples is available at the following link: dev.ti.com C2000Ware Examples.

#### 12.4.1.1 Buffered DAC Enable

FILE: buffdac_ex1_enable.c

This example generates a voltage on the buffered DAC output, DACOUTA/ADCINA0 and uses the default DAC reference setting of VDAC.

*ExternalConnections*

- When the DAC reference is set to VDAC, an external reference voltage must be applied to the VDAC pin. This can be accomplished by connecting a jumper wire from 3.3V to ADCINB0.

#### 12.4.1.2 Buffered DAC Random

FILE: buffdac_ex2_random.c

This example generates random voltages on the buffered DAC output, DACOUTA/ADCINA0 and uses the default DAC reference setting of VDAC.

*ExternalConnections*

- When the DAC reference is set to VDAC, an external reference voltage must be applied to the VDAC pin. This can accomplished by connecting a jumper wire from 3.3V to ADCINB0.

#### 12.4.1.3 Buffered DAC Sine (buffdac_sine)

FILE: buffdac_ex3_waveform.c

This example generates a sine wave on the buffered DAC output, DACOUTA/ADCINA0 (HSEC Pin 9) and uses the default DAC reference setting of VDAC.

When the DAC reference is set to VDAC, an external reference voltage must be applied to the VDAC pin. This can accomplished by connecting a jumper wire from 3.3V to ADCINB0.

## 12.5 DAC Registers

This section describes the Buffered Digital to Analog Converter registers.

### *12.5.1 DAC Base Addresses*

**Table 12-1. DAC Base Address Table**

| Device Registers | Register Name | Start Address | End Address |
|---|---|---|---|
| DacaRegs | DAC_REGS | 0x0000_5C00 | 0x0000_5C0F |
| DacbRegs | DAC_REGS | 0x0000_5C10 | 0x0000_5C1F |
| DaccRegs | DAC_REGS | 0x0000_5C20 | 0x0000_5C2F |



Copyright © 2024 Texas Instruments Incorporated

### 12.5.2 DAC_REGS Registers

Table 12-2 lists the memory-mapped registers for the DAC_REGS registers. All register offset addresses not listed in Table 12-2 should be considered as reserved locations and the register contents should not be modified.

**Table 12-2. DAC_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | DACREV | DAC Revision Register | | Go |
| 1h | DACCTL | DAC Control Register | EALLOW | Go |
| 2h | DACVALA | DAC Value Register - Active | | Go |
| 3h | DACVALS | DAC Value Register - Shadow | | Go |
| 4h | DACOUTEN | DAC Output Enable Register | EALLOW | Go |
| 5h | DACLOCK | DAC Lock Register | EALLOW | Go |
| 6h | DACTRIM | DAC Trim Register | EALLOW | Go |

Complex bit access types are encoded to fit into small table cells. Table 12-3 shows the codes that are used for access types in this section.

**Table 12-3. DAC_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| R-0 | R<br>-0 | Read<br>Returns 0s |
| Write Type | | |
| W | W | Write |
| WSonce | W<br>Sonce | Write<br>Set once |
| Reset or Default Value | | |
| *-n* | | Value after reset or the default value |



Copyright © 2024 Texas Instruments Incorporated

#### 12.5.2.1 DACREV Register (Offset = 0h) [Reset = 0000h]

DACREV is shown in Figure 12-2 and described in Table 12-4.

Return to the Summary Table.

DAC Revision Register

**Figure 12-2. DACREV Register**

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

**Table 12-4. DACREV Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R | 0h | Reserved |
| 7-0 | REV | R | 0h | DAC Revision<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 12.5.2.2 DACCTL Register (Offset = 1h) [Reset = 0000h]

DACCTL is shown in Figure 12-3 and described in Table 12-5.

Return to the Summary Table.

DAC Control Register

**Figure 12-3. DACCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SYNCSEL | | | | RESERVED | LOADMODE | RESERVED | DACREFSEL |
| R/W-0h | | | | R-0h | R/W-0h | R-0h | R/W-0h |

**Table 12-5. DACCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R | 0h | Reserved |
| 7-4 | SYNCSEL | R/W | 0h | DAC EPWMSYNCPER select. Determines which EPWMSYNCPER signal will update the DACVALA register.<br>Where n represents the maximum number of EPWMSYNCPER signals available on the device:<br>0 EPWM1SYNCPER<br>1 EPWM2SYNCPER<br>2 EPWM3SYNCPER<br>...<br>n-1 EPWMnSYNCPER<br>Reset type: SYSRSn |
| 3 | RESERVED | R | 0h | Reserved |
| 2 | LOADMODE | R/W | 0h | DACVALA load mode. Determines when the DACVALA register is updated with the value from DACVALS.<br>0 Load on next SYSCLK<br>1 Load on next EPWMSYNCPER specified by SYNCSEL<br>Reset type: SYSRSn |
| 1 | RESERVED | R | 0h | Reserved |
| 0 | DACREFSEL | R/W | 0h | DAC reference select. Selects which voltage references are used by the DAC.<br>0 VDAC/VSSA are the reference voltages<br>1 ADC VREFHI/VSSA are the reference voltages<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 12.5.2.3 DACVALA Register (Offset = 2h) [Reset = 0000h]

DACVALA is shown in Figure 12-4 and described in Table 12-6.

Return to the Summary Table.

DAC Value Register - Active

**Figure 12-4. DACVALA Register**

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

**Table 12-6. DACVALA Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | RESERVED | R | 0h | Reserved |
| 11-0 | DACVALA | R | 0h | Active output code currently driven by the DAC<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 12.5.2.4 DACVALS Register (Offset = 3h) [Reset = 0000h]

DACVALS is shown in Figure 12-5 and described in Table 12-7.

Return to the Summary Table.

DAC Value Register - Shadow

**Figure 12-5. DACVALS Register**

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

**Table 12-7. DACVALS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | RESERVED | R | 0h | Reserved |
| 11-0 | DACVALS | R/W | 0h | Shadow output code to be loaded into DACVALA<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 12.5.2.5 DACOUTEN Register (Offset = 4h) [Reset = 0000h]

DACOUTEN is shown in Figure 12-6 and described in Table 12-8.

Return to the Summary Table.

DAC Output Enable Register

**Figure 12-6. DACOUTEN Register**

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

**Table 12-8. DACOUTEN Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-1 | RESERVED | R | 0h | Reserved |
| 0 | DACOUTEN | R/W | 0h | DAC output enable<br>0 DAC output is disabled<br>1 DAC output is enabled<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 12.5.2.6 DACLOCK Register (Offset = 5h) [Reset = 0000h]

DACLOCK is shown in Figure 12-7 and described in Table 12-9.

Return to the Summary Table.

DAC Lock Register

**Figure 12-7. DACLOCK Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| KEY | | | | RESERVED | | | |
| R-0/W-0h | | | | R-0h | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | DACOUTEN | DACVAL | DACCTL |
| R-0h | | | | | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h |

**Table 12-9. DACLOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | KEY | R-0/W | 0h | Writes to this register succeed only if this field is written with a value of 0xA.<br>Only 16-bit writes will succeed (provided the KEY matches). Read-modify-writes to individual bits in this register will be ignored.<br>Reset type: SYSRSn |
| 11-3 | RESERVED | R | 0h | Reserved |
| 2 | DACOUTEN | R/WSonce | 0h | Lock write-access to the DACOUTEN register.<br>0 DACOUTEN register is not locked. Write 0 to this bit has no effect.<br>1 DACOUTEN register is locked. Only a system reset can clear this bit.<br>Reset type: SYSRSn |
| 1 | DACVAL | R/WSonce | 0h | Lock write-access to the DACVALS register.<br>0 DACVALS register is not locked. Write 0 to this bit has no effect.<br>1 DACVALS register is locked. Only a system reset can clear this bit.<br>Reset type: SYSRSn |
| 0 | DACCTL | R/WSonce | 0h | Lock write-access to the DACCTL register.<br>0 DACCTL register is not locked. Write 0 to this bit has no effect.<br>1 DACCTL register is locked. Only a system reset can clear this bit.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

#### 12.5.2.7 DACTRIM Register (Offset = 6h) [Reset = 0000h]

DACTRIM is shown in Figure 12-8 and described in Table 12-10.

Return to the Summary Table.

DAC Trim Register

**Figure 12-8. DACTRIM Register**

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

**Table 12-10. DACTRIM Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | RESERVED | R | 0h | Reserved |
| 11-8 | RESERVED | R/W | 0h | Reserved |
| 7-0 | OFFSET_TRIM | R/W | 0h | DAC Offset Trim.<br>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.<br>Reset type: SYSRSn |

### *12.5.3 DAC Registers to Driverlib Functions*

**Table 12-11. DAC Registers to Driverlib Functions**

| File | Driverlib Function |
|---|---|
| **DACREV** | |
| dac.h | DAC_getRevision |
| **DACCTL** | |
| dac.h | DAC_setReferenceVoltage |
| dac.h | DAC_setLoadMode |
| dac.h | DAC_setPWMSyncSignal |
| **DACVALA** | |
| dac.h | DAC_getActiveValue |
| **DACVALS** | |
| dac.h | DAC_setShadowValue |
| dac.h | DAC_getShadowValue |
| **DACOUTEN** | |
| dac.h | DAC_enableOutput |
| dac.h | DAC_disableOutput |
| **DACLOCK** | |
| dac.h | DAC_lockRegister |
| dac.h | DAC_isRegisterLocked |
| **DACTRIM** | |
| dac.c | DAC_tuneOffsetTrim |
| dac.h | DAC_setOffsetTrim |



Copyright © 2024 Texas Instruments Incorporated

**Table 12-11. DAC Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| dac.h | DAC_getOffsetTrim |



Copyright © 2024 Texas Instruments Incorporated

Chapter 13

# 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.

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.


**13.1 Introduction** ..... 1804
**13.2 Comparator** ..... 1805
**13.3 Reference DAC** ..... 1806
**13.4 Ramp Generator** ..... 1807
**13.5 Digital Filter** ..... 1810
**13.6 Using the CMPSS** ..... 1811
**13.7 Software** ..... 1813
**13.8 CMPSS Registers** ..... 1814





Copyright © 2024 Texas Instruments Incorporated

## 13.1 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. 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. 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.

### *13.1.1 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*

### *13.1.2 Features*

Each CMPSS includes:

- Two analog comparators
- Two programmable reference 12-bit DACs
- One decrementing ramp generator
- Two digital filters, max filter clock prescale = $2^{10}$
- Ability to synchronize submodules with EPWMSYNCPER
- 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
- The DAC reference voltage can be set to VDDA or VDAC



Copyright © 2024 Texas Instruments Incorporated

### 13.1.3 Block Diagram

The block diagram for the CMPSS is shown in Figure 13-1.

- 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.

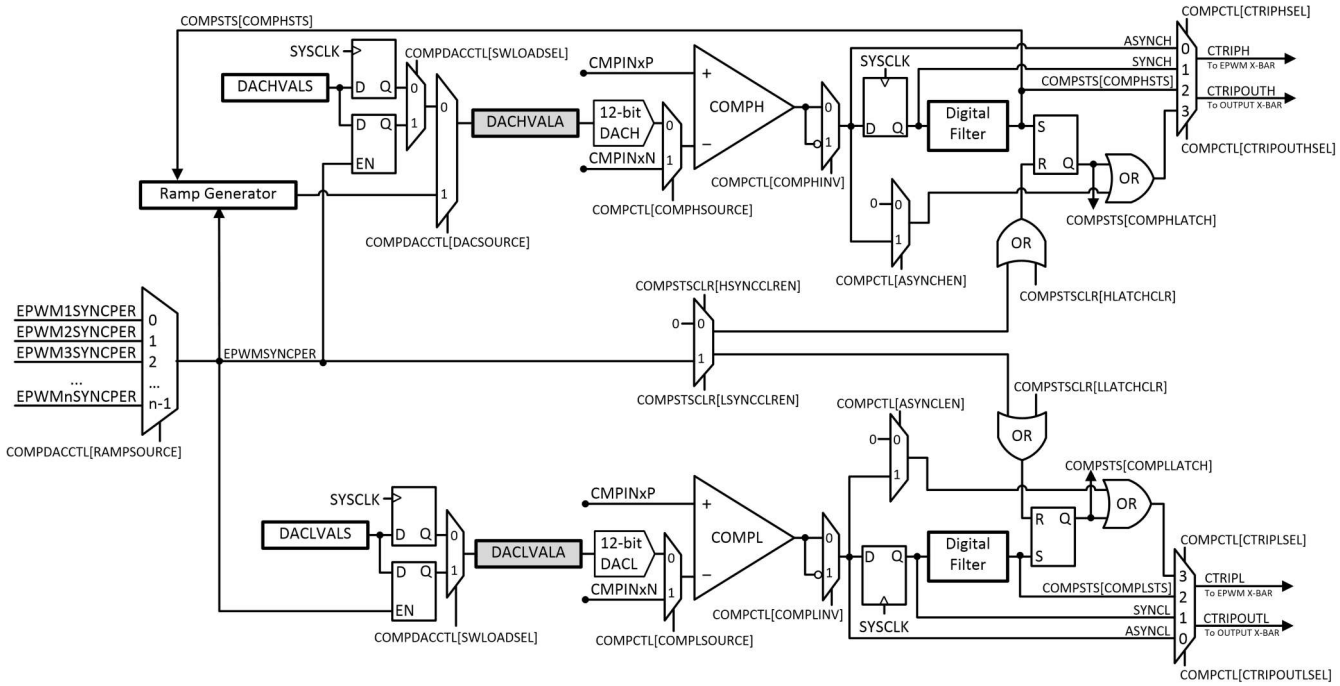



**Figure 13-1. CMPSS Module Block Diagram**

## 13.2 Comparator

The comparator generates a high digital output when the voltage on the positive input is greater than the voltage on the negative input, and a low digital output when the voltage on the positive input is less than the voltage on the negative input. The comparator is illustrated in Figure 13-2.

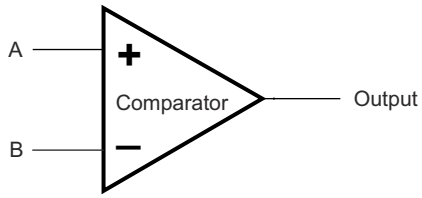



**Figure 13-2. Comparator Block Diagram**

| Voltages | Output |
|---|---|
| Voltage A > Voltage B | 1 |
| Voltage A < Voltage B | 0 |



Copyright © 2024 Texas Instruments Incorporated

## 13.3 Reference DAC

Each reference 12-bit DAC can be configured to drive a reference voltage into the negative input of the respective comparator. The reference 12-bit DAC output is internal only and cannot be observed externally.

Two sets of DACxVAL registers, DACxVALA and DACxVALS, are present for each reference 12-bit DAC. DACxVALA is a read-only register that actively controls the reference 12-bit DAC value. DACxVALS is a writable shadow register that loads into DACxVALA either immediately or synchronized with the next EPWMSYNCPER event. The high 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. The high-voltage reference is VDDA by default, but the high voltage reference can be configured to be VDAC using the COMPDACCTL register. The reference 12-bit DAC is illustrated in Figure 13-3.

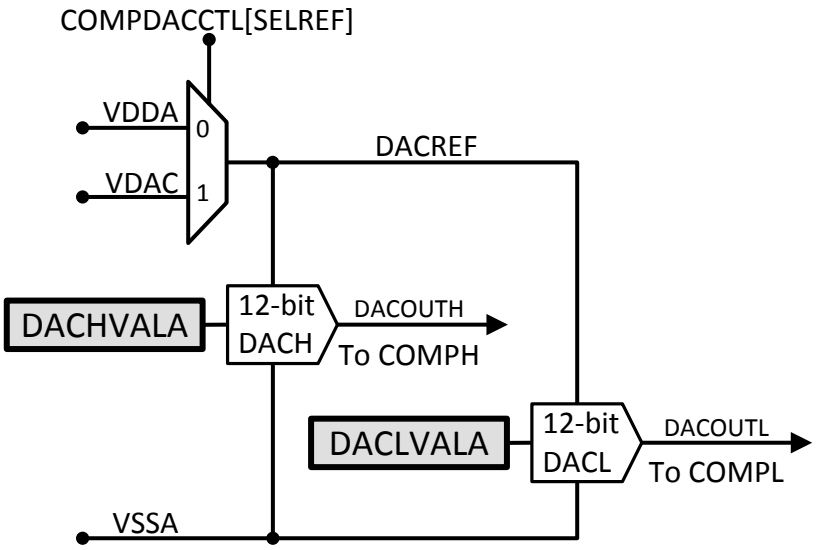



**Figure 13-3. Reference DAC Block Diagram**

The output of the reference 12-bit DAC can be calculated as:

$$DACOUT = \frac{(1 + DACVALA)*DACREF}{4096} \quad (16)$$

**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.



Copyright © 2024 Texas Instruments Incorporated

## 13.4 Ramp Generator

This section discusses the characteristics and behavior of the ramp generator.

### *13.4.1 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 RAMPDECVALA.

The ramp generator is enabled by setting DACSOURCE = 1. When DACSOURCE = 1 is selected, the value of RAMPSTS is loaded from RAMPMAXREFS and the register remains static until the selected EPWMSYNCPER signal is received. After receiving the selected EPWMSYNCPER signal, the value of RAMPDECVALA is subtracted from RAMPSTS on every subsequent SYSCLK cycle.

To prevent the subtraction from commencing a SYSCLK cycle after a EPWMSYNCPER event, the RAMPDLYA register 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.



Copyright © 2024 Texas Instruments Incorporated

### 13.4.2 Ramp Generator Behavior

The ramp generator makes state changes on every rising edge of DACSOURCE, EPWMSYNCPER, and COMPHSTS.

On the rising edge of DACSOURCE: RAMPMAXREFA, RAMPDECVALA, and RAMPDLYA are loaded with the shadow registers. RAMPSTS is loaded with RAMPMAXREFS.

On the rising edge of the selected EPWMSYNCPER: RAMPMAXREFA, RAMPDECVALA, and RAMPDLYA are loaded with the shadow registers. RAMPSTS is loaded with RAMPMAXREFS and starts decrementing when RAMPDLYA counter reaches zero.

On the rising edge of COMPHSTS with RAMPLOADSEL = 1: RAMPMAXREFA, RAMPDECVALA, and RAMPDLYA are loaded with the shadow registers. RAMPSTS is loaded with RAMPMAXREFS and stops decrementing.

On the rising edge of COMPHSTS with RAMPLOADSEL = 0: RAMPSTS is loaded with RAMPMAXREFA and stops decrementing.

Additionally, if the value of RAMPSTS reaches zero, the RAMPSTS register remains static at zero until the next EPWMSYNCPER is received. These state changes are illustrated in the ramp generator block diagram in Figure 13-4.

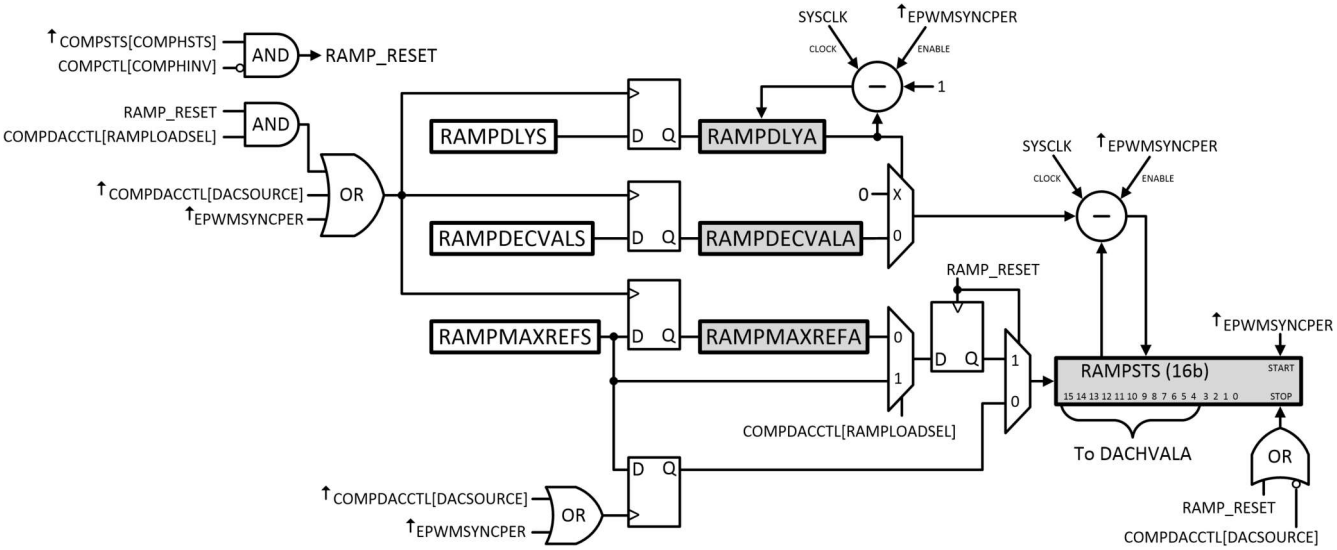



**Figure 13-4. Ramp Generator Block Diagram**



Copyright © 2024 Texas Instruments Incorporated

### 13.4.3 Ramp Generator Behavior at Corner Cases

Since the ramp generator makes state changes on every rising edge of EPWMSYNCPER and COMPHSTS, the following behavior can be expected on instances when these two events occur simultaneously or very close together.

Case 1: COMPHSTS rising edge occurs one or more cycles before EPWMSYNCPER rising edge.
RAMPSTS stops decrementing on COMPHSTS rising edge event. RAMPSTS starts decrementing on EPWMSYNCPER rising edge event when RAMPDLYA reaches 0.

Case 2: COMPHSTS rising edge occurs simultaneously as EPWMSYNCPER rising edge.
EPWMSYNCPER rising edge event takes precedence and RAMPSTS starts decrementing when RAMPDLYA reaches 0. COMPHSTS rising edge event is ignored and does not halt RAMPSTS.

Case 2: COMPHSTS rising edge occurs simultaneously as EPWMSYNCPER rising edge.
COMPHSTS rising edge event takes precedence and RAMPSTS stops decrementing. EPWMSYNCPER rising edge event is ignored and does not start decrementing RAMPSTS when RAMPDLYA reaches 0.

Case 3: COMPHSTS rising edge occurs one or more cycles after EPWMSYNCPER rising edge but before RAMPDLYA reaches 0.
RAMPSTS does not decrement when RAMPDLYA reaches 0.

Case 4: COMPHSTS rising edge occurs simultaneously as RAMPDLYA reaches 0 from EPWMSYNCPER rising edge.
RAMPSTS does not decrement.

This behavior is also illustrated in Figure 13-5.

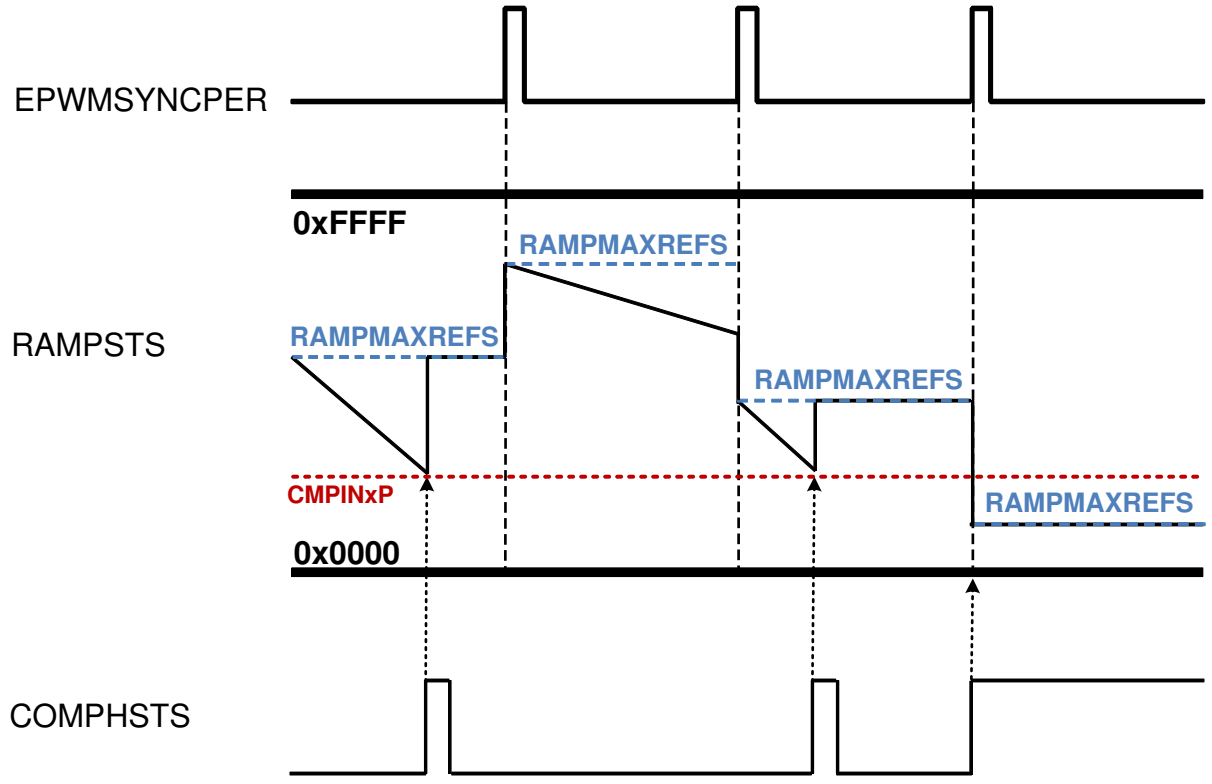



**Figure 13-5. Ramp Generator Behavior**



Copyright © 2024 Texas Instruments Incorporated

## 13.5 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 13-6.

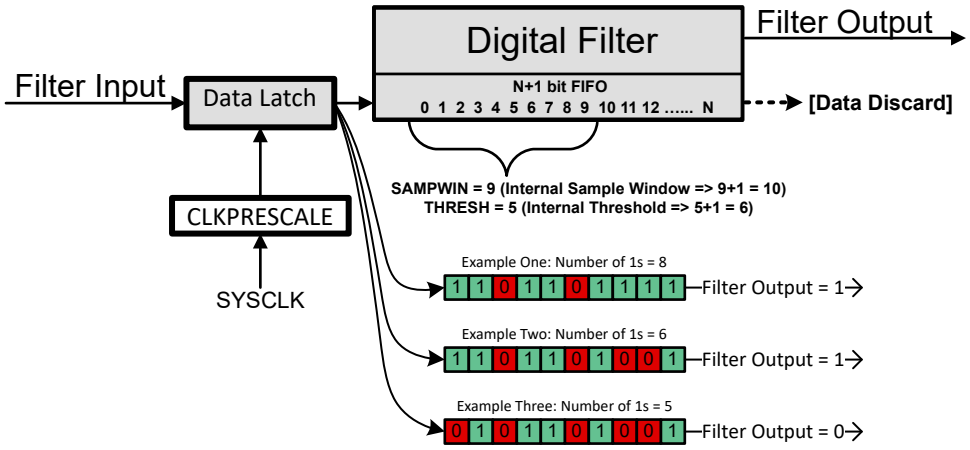



**Figure 13-6. Digital Filter Behavior**

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;
    }
}
```



Copyright © 2024 Texas Instruments Incorporated

#### 13.5.1 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.

### 13.6 Using the CMPSS

#### 13.6.1 LATCHCLR and EPWMSYNCPER Signals

The LATCHCLR signal holds 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.

EPWMSYNCPER comes from the Time-Base submodule of the EPWM. For a detailed description of how this signal is generated, refer to the Time-Base submodule 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 EPWMSYNCI and EPWMSYNCO. For a description of what are these signals, see the *Enhanced Pulse Width Modulator (ePWM)* chapter.

#### 13.6.2 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.



Copyright © 2024 Texas Instruments Incorporated

### 13.6.3 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, 0xFFF.
   - 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.

### 13.6.4 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.



Copyright © 2024 Texas Instruments Incorporated

## 13.7 Software

### *13.7.1 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.

#### 13.7.1.1 CMPSS Asynchronous Trip

FILE: cmpss_ex1_asynch.c

This example enables the CMPSS1 COMPH comparator and feeds the asynchronous CTRIPOUTH signal to the GPIO14/OUTPUTXBAR3 pin and CTRIPH to GPIO15/EPWM8B.

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 GPIO15 and is configured to be tripped by CTRIPOUTH.

When a low input(VSS) is provided to CMPIN1P,
- Trip signal(GPIO14) output is low
- PWM8B(GPIO15) gives a PWM signal

When a high input(higher than VDD/2) is provided to CMPIN1P,
- Trip signal(GPIO14) output turns high
- PWM8B(GPIO15) gets tripped and outputs as high

*External Connections*
- Outputs can be observed on GPIO14 and GPIO15
- Comparator input pin is on ADCINA2

*Watch Variables*
- None

#### 13.7.1.2 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 GPIO14/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,
- GPIO14 output is low

When a high input(higher than VDD/2) is provided to CMPIN1P,
- GPIO14 output turns high



Copyright © 2024 Texas Instruments Incorporated

## 13.8 CMPSS Registers

This section describes the CMPSS Registers.

### *13.8.1 CMPSS Base Addresses*

**Table 13-1. CMPSS Base Address Table**

| Device Registers | Register Name | Start Address | End Address |
|---|---|---|---|
| Cmpss1Regs | CMPSS_REGS | 0x0000_5C80 | 0x0000_5C9F |
| Cmpss2Regs | CMPSS_REGS | 0x0000_5CA0 | 0x0000_5CBF |
| Cmpss3Regs | CMPSS_REGS | 0x0000_5CC0 | 0x0000_5CDF |
| Cmpss4Regs | CMPSS_REGS | 0x0000_5CE0 | 0x0000_5CFF |
| Cmpss5Regs | CMPSS_REGS | 0x0000_5D00 | 0x0000_5D1F |
| Cmpss6Regs | CMPSS_REGS | 0x0000_5D20 | 0x0000_5D3F |
| Cmpss7Regs | CMPSS_REGS | 0x0000_5D40 | 0x0000_5D5F |
| Cmpss8Regs | CMPSS_REGS | 0x0000_5D60 | 0x0000_5D7F |



Copyright © 2024 Texas Instruments Incorporated

### 13.8.2 CMPSS_REGS Registers

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

**Table 13-2. CMPSS_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | COMPCTL | CMPSS Comparator Control Register | EALLOW | Go |
| 1h | COMPHYSCTL | CMPSS Comparator Hysteresis Control Register | EALLOW | Go |
| 2h | COMPSTS | CMPSS Comparator Status Register | | Go |
| 3h | COMPSTSCLR | CMPSS Comparator Status Clear Register | EALLOW | Go |
| 4h | COMPDACCTL | CMPSS DAC Control Register | EALLOW | Go |
| 6h | DACHVALS | CMPSS High DAC Value Shadow Register | | Go |
| 7h | DACHVALA | CMPSS High DAC Value Active Register | | Go |
| 8h | RAMPMAXREFA | CMPSS Ramp Max Reference Active Register | | Go |
| Ah | RAMPMAXREFS | CMPSS Ramp Max Reference Shadow Register | | Go |
| Ch | RAMPDECVALA | CMPSS Ramp Decrement Value Active Register | | Go |
| Eh | RAMPDECVALS | CMPSS Ramp Decrement Value Shadow Register | | Go |
| 10h | RAMPSTS | CMPSS Ramp Status Register | | Go |
| 12h | DACLVALS | CMPSS Low DAC Value Shadow Register | | Go |
| 13h | DACLVALA | CMPSS Low DAC Value Active Register | | Go |
| 14h | RAMPDLYA | CMPSS Ramp Delay Active Register | | Go |
| 15h | RAMPDLYS | CMPSS Ramp Delay Shadow Register | | Go |
| 16h | CTRIPLFILCTL | CTRIPL Filter Control Register | EALLOW | Go |
| 17h | CTRIPLFILCLKCTL | CTRIPL Filter Clock Control Register | EALLOW | Go |
| 18h | CTRIPHFILCTL | CTRIPH Filter Control Register | EALLOW | Go |
| 19h | CTRIPHFILCLKCTL | CTRIPH Filter Clock Control Register | EALLOW | Go |
| 1Ah | COMPLOCK | CMPSS Lock Register | EALLOW | Go |

Complex bit access types are encoded to fit into small table cells. Table 13-3 shows the codes that are used for access types in this section.

**Table 13-3. CMPSS_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| R-0 | R<br>-0 | Read<br>Returns 0s |
| Write Type | | |
| W | W | Write |
| W1S | W<br>1S | Write<br>1 to set |
| WSonce | W<br>Sonce | Write<br>Set once |
| Reset or Default Value | | |
| *-n* | | Value after reset or the default value |
| Register Array Variables | | |



Copyright © 2024 Texas Instruments Incorporated

**Table 13-3. CMPSS_REGS Access Type Codes (continued)**

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



Copyright © 2024 Texas Instruments Incorporated

### 13.8.2.1 COMPCTL Register (Offset = 0h) [Reset = 0000h]

COMPCTL is shown in Figure 13-7 and described in Table 13-4.

Return to the Summary Table.

CMPSS Comparator Control Register

**Figure 13-7. COMPCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| COMPDACE | ASYNCLEN | CTRIPOUTLSEL | | CTRIPLSEL | | COMPLINV | COMPLSOURCE |
| R/W-0h | R/W-0h | R/W-0h | | R/W-0h | | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | ASYNCHEN | CTRIPOUTHSEL | | CTRIPHSEL | | COMPHINV | COMPHSOURCE |
| R-0h | R/W-0h | R/W-0h | | R/W-0h | | R/W-0h | R/W-0h |

**Table 13-4. COMPCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | COMPDACE | R/W | 0h | Comparator/DAC enable.<br>0 Comparator/DAC disabled<br>1 Comparator/DAC enabled<br>Reset type: SYSRSn |
| 14 | ASYNCLEN | 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.<br>0 Asynchronous comparator output does not feed into OR gate with latched digital filter output<br>1 Asynchronous comparator output feeds into OR gate with latched digital filter output<br>Reset type: SYSRSn |
| 13-12 | CTRIPOUTLSEL | R/W | 0h | Low comparator CTRIPOUTL source select.<br>0 Asynchronous comparator output drives CTRIPOUTL<br>1 Synchronous comparator output drives CTRIPOUTL<br>2 Output of digital filter drives CTRIPOUTL<br>3 Latched output of digital filter drives CTRIPOUTL<br>Reset type: SYSRSn |
| 11-10 | CTRIPLSEL | R/W | 0h | Low comparator CTRIPL source select.<br>0 Asynchronous comparator output drives CTRIPL<br>1 Synchronous comparator output drives CTRIPL<br>2 Output of digital filter drives CTRIPL<br>3 Latched output of digital filter drives CTRIPL<br>Reset type: SYSRSn |
| 9 | COMPLINV | R/W | 0h | Low comparator output invert.<br>0 Output of comparator is not inverted<br>1 Output of comparator is inverted<br>Reset type: SYSRSn |
| 8 | COMPLSOURCE | R/W | 0h | Low comparator input source.<br>0 Inverting input of comparator driven by internal DAC<br>1 Inverting input of comparator driven through external pin<br>Reset type: SYSRSn |
| 7 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 13-4. 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.<br>0 Asynchronous comparator output does not feed into OR gate with latched digital filter output<br>1 Asynchronous comparator output feeds into OR gate with latched digital filter output<br>Reset type: SYSRSn |
| 5-4 | CTRIPOUTHSEL | R/W | 0h | High comparator CTRIPOUTH source select.<br>0 Asynchronous comparator output drives CTRIPOUTH<br>1 Synchronous comparator output drives CTRIPOUTH<br>2 Output of digital filter drives CTRIPOUTH<br>3 Latched output of digital filter drives CTRIPOUTH<br>Reset type: SYSRSn |
| 3-2 | CTRIPHSEL | R/W | 0h | High comparator CTRIPH source select.<br>0 Asynchronous comparator output drives CTRIPH<br>1 Synchronous comparator output drives CTRIPH<br>2 Output of digital filter drives CTRIPH<br>3 Latched output of digital filter drives CTRIPH<br>Reset type: SYSRSn |
| 1 | COMPHINV | R/W | 0h | High comparator output invert.<br>0 Output of comparator is not inverted<br>1 Output of comparator is inverted<br>Reset type: SYSRSn |
| 0 | COMPHSOURCE | R/W | 0h | High comparator input source.<br>0 Inverting input of comparator driven by internal DAC<br>1 Inverting input of comparator driven through external pin<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 13.8.2.2 COMPHYSCTL Register (Offset = 1h) [Reset = 0000h]

COMPHYSCTL is shown in Figure 13-8 and described in Table 13-5.

Return to the Summary Table.

CMPSS Comparator Hysteresis Control Register

**Figure 13-8. 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 13-5. COMPHYSCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-3 | RESERVED | R | 0h | Reserved |
| 2-0 | COMPHYS | R/W | 0h | Comparator hysteresis. Sets the amount of hysteresis on the comparator inputs.<br>0 None<br>1 Set to typical hysteresis<br>2 Set to 2x of typical hysteresis<br>3 Set to 3x of typical hysteresis<br>4 Set to 4x of typical hysteresis<br>others : undefined<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 13.8.2.3 COMPSTS Register (Offset = 2h) [Reset = 0000h]

COMPSTS is shown in Figure 13-9 and described in Table 13-6.

Return to the Summary Table.

CMPSS Comparator Status Register

**Figure 13-9. 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 13-6. COMPSTS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-10 | RESERVED | R | 0h | Reserved |
| 9 | COMPLLATCH | R | 0h | Latched value of low comparator digital filter output<br>Reset type: SYSRSn |
| 8 | COMPLSTS | R | 0h | Low comparator digital filter output<br>Reset type: SYSRSn |
| 7-2 | RESERVED | R | 0h | Reserved |
| 1 | COMPHLATCH | R | 0h | Latched value of high comparator digital filter output<br>Reset type: SYSRSn |
| 0 | COMPHSTS | R | 0h | High comparator digital filter output<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 13.8.2.4 COMPSTSCLR Register (Offset = 3h) [Reset = 0000h]

COMPSTSCLR is shown in Figure 13-10 and described in Table 13-7.

Return to the Summary Table.

CMPSS Comparator Status Clear Register

**Figure 13-10. 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 13-7. COMPSTSCLR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-11 | RESERVED | R | 0h | Reserved |
| 10 | LSYNCCLREN | R/W | 0h | Low comparator latch EPWMSYNCPER clear. Enable EPWMSYNCPER reset of low comparator digital filter output latch COMPSTS[COMPLLATCH].<br>0 EPWMSYNCPER will not reset latch<br>1 EPWMSYNCPER will reset latch<br>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.<br>0 No effect<br>1 Generate a single pulse of latch reset signal for COMPSTS[COMPLLATCH]<br>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].<br>0 EPWMSYNCPER will not reset latch<br>1 EPWMSYNCPER will reset latch<br>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.<br>0 No effect<br>1 Generate a single pulse of latch reset signal for COMPSTS[COMPHLATCH]<br>Reset type: SYSRSn |
| 0 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 13.8.2.5 COMPDACCTL Register (Offset = 4h) [Reset = 0000h]

COMPDACCTL is shown in Figure 13-11 and described in Table 13-8.

Return to the Summary Table.

CMPSS DAC Control Register

**Figure 13-11. COMPDACCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| FREESOFT | | RESERVED | | | | | |
| R/W-0h | | R-0h | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SWLOADSEL | RAMPLOADSEL | SELREF | RAMPSOURCE | | | | DACSOURCE |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | | | | R/W-0h |

**Table 13-8. COMPDACCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-14 | FREESOFT | R/W | 0h | Free-run or software-run emulation behavior. Behavior of the ramp generator during emulation suspend.<br>00b Ramp generator stops immediately during emulation suspend<br>01b Ramp generator completes current ramp and stops at next EPWMSYNCPER during emulation suspend<br>1Xb Ramp generator runs freely<br>Reset type: SYSRSn |
| 13-8 | RESERVED | R | 0h | Reserved |
| 7 | SWLOADSEL | R/W | 0h | Software load select. Determines whether DACxVALA is updated from DACxVALS on SYSCLK or EPWMSYNCPER.<br>0 DACxVALA is updated from DACxVALS on SYSCLK<br>1 DACxVALA is updated from DACxVALS on EPWMSYNCPER<br>Reset type: SYSRSn |
| 6 | RAMPLOADSEL | R/W | 0h | Ramp load select. Determines whether RAMPSTS is updated from RAMPMAXREFA or RAMPMAXREFS when COMPSTS[COMPHSTS] is triggered.<br>0 RAMPSTS is loaded from RAMPMAXREFA<br>1 RAMPSTS is loaded from RAMPMAXREFS<br>Reset type: SYSRSn |
| 5 | SELREF | R/W | 0h | DAC reference select. Determines which voltage supply is used as the reference for the internal comparator DACs.<br>0 VDDA is the voltage reference for the DAC<br>1 VDAC is the voltage reference for the DAC<br>Reset type: SYSRSn |
| 4-1 | RAMPSOURCE | R/W | 0h | Ramp generator source select. Determines which EPWMSYNCPER signal is used within the CMPSS module.<br>Where n represents the maximum number of EPWMSYNCPER signals available on the device:<br>0 EPWM1SYNCPER<br>1 EPWM2SYNCPER<br>2 EPWM3SYNCPER<br>...<br>n-1 EPWMnSYNCPER<br>Reset type: SYSRSn |
| 0 | DACSOURCE | R/W | 0h | DAC source select. Determines whether DACHVALA is updated from DACHVALS or from the ramp generator.<br>0 DACHVALA is updated from DACHVALS<br>1 DACHVALA is updated from the ramp generator<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 13.8.2.6 DACHVALS Register (Offset = 6h) [Reset = 0000h]

DACHVALS is shown in Figure 13-12 and described in Table 13-9.

Return to the Summary Table.

CMPSS High DAC Value Shadow Register

**Figure 13-12. 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 13-9. DACHVALS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | RESERVED | R | 0h | Reserved |
| 11-0 | DACVAL | R/W | 0h | High DAC shadow value. When COMPDACCTL[DACSOURCE]=0, the value of DACHVALS is loaded into DACHVALA on the trigger signal selected by COMPDACCTL[SWLOADSEL].<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 13.8.2.7 DACHVALA Register (Offset = 7h) [Reset = 0000h]

DACHVALA is shown in Figure 13-13 and described in Table 13-10.

Return to the Summary Table.

CMPSS High DAC Value Active Register

**Figure 13-13. 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 13-10. DACHVALA Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | RESERVED | R | 0h | Reserved |
| 11-0 | DACVAL | R | 0h | High DAC active value. Value that is actively driven by the high DAC.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 13.8.2.8 RAMPMAXREFA Register (Offset = 8h) [Reset = 0000h]

RAMPMAXREFA is shown in Figure 13-14 and described in Table 13-11.

Return to the Summary Table.

CMPSS Ramp Max Reference Active Register

**Figure 13-14. RAMPMAXREFA Register**

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

**Table 13-11. RAMPMAXREFA Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RAMPMAXREF | R | 0h | Ramp maximum reference active value. Latched value to be loaded into ramp generator RAMPSTS.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 13.8.2.9 RAMPMAXREFS Register (Offset = Ah) [Reset = 0000h]

RAMPMAXREFS is shown in Figure 13-15 and described in Table 13-12.

Return to the Summary Table.

CMPSS Ramp Max Reference Shadow Register

**Figure 13-15. RAMPMAXREFS Register**

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

**Table 13-12. RAMPMAXREFS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RAMPMAXREF | R/W | 0h | Ramp maximum reference shadow. Unlatched value to be loaded into ramp generator RAMPSTS.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 13.8.2.10 RAMPDECVALA Register (Offset = Ch) [Reset = 0000h]

RAMPDECVALA is shown in Figure 13-16 and described in Table 13-13.

Return to the Summary Table.

CMPSS Ramp Decrement Value Active Register

**Figure 13-16. RAMPDECVALA Register**

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

**Table 13-13. RAMPDECVALA Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RAMPDECVAL | R | 0h | Ramp decrement value active. Latched value that will be subtracted from RAMPSTS.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 13.8.2.11 RAMPDECVALS Register (Offset = Eh) [Reset = 0000h]

RAMPDECVALS is shown in Figure 13-17 and described in Table 13-14.

Return to the Summary Table.

CMPSS Ramp Decrement Value Shadow Register

**Figure 13-17. RAMPDECVALS Register**

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

**Table 13-14. RAMPDECVALS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RAMPDECVAL | R/W | 0h | Ramp decrement value shadow. Unlatched value to be loaded into RAMPDECVALA.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 13.8.2.12 RAMPSTS Register (Offset = 10h) [Reset = 0000h]

RAMPSTS is shown in Figure 13-18 and described in Table 13-15.

Return to the Summary Table.

CMPSS Ramp Status Register

**Figure 13-18. RAMPSTS Register**

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

**Table 13-15. RAMPSTS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RAMPVALUE | R | 0h | Ramp value. Present value of ramp generator.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 13.8.2.13 DACLVALS Register (Offset = 12h) [Reset = 0000h]

DACLVALS is shown in Figure 13-19 and described in Table 13-16.

Return to the Summary Table.

CMPSS Low DAC Value Shadow Register

**Figure 13-19. 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 13-16. DACLVALS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | RESERVED | R | 0h | Reserved |
| 11-0 | DACVAL | R/W | 0h | Low DAC shadow value. value to be loaded into DACLVALA on the trigger signal selected by COMPDACCTL[SWLOADSEL].<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 13.8.2.14 DACLVALA Register (Offset = 13h) [Reset = 0000h]

DACLVALA is shown in Figure 13-20 and described in Table 13-17.

Return to the Summary Table.

CMPSS Low DAC Value Active Register

**Figure 13-20. 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 13-17. DACLVALA Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | RESERVED | R | 0h | Reserved |
| 11-0 | DACVAL | R | 0h | Low DAC active value. Value that is actively driven by the low DAC.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 13.8.2.15 RAMPDLYA Register (Offset = 14h) [Reset = 0000h]

RAMPDLYA is shown in Figure 13-21 and described in Table 13-18.

Return to the Summary Table.

CMPSS Ramp Delay Active Register

**Figure 13-21. RAMPDLYA Register**

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

**Table 13-18. RAMPDLYA Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-13 | RESERVED | R | 0h | Reserved |
| 12-0 | DELAY | R | 0h | Ramp delay active value. Latched value of the number of cycles to delay the start of the ramp generator decrementer after a EPWMSYNCPER is received.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 13.8.2.16 RAMPDLYS Register (Offset = 15h) [Reset = 0000h]

RAMPDLYS is shown in Figure 13-22 and described in Table 13-19.

Return to the Summary Table.

CMPSS Ramp Delay Shadow Register

**Figure 13-22. RAMPDLYS Register**

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

**Table 13-19. RAMPDLYS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-13 | RESERVED | R | 0h | Reserved |
| 12-0 | DELAY | R/W | 0h | Ramp delay shadow value. Unlatched value to be loaded into RAMPDLYA.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 13.8.2.17 CTRIPLFILCTL Register (Offset = 16h) [Reset = 0000h]

CTRIPLFILCTL is shown in Figure 13-23 and described in Table 13-20.

Return to the Summary Table.

CTRIPL Filter Control Register

**Figure 13-23. CTRIPLFILCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| FILINIT | RESERVED | THRESH | | | | | SAMPWIN |
| R-0/W1S-0h | R-0h | R/W-0h | | | | | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SAMPWIN | | | | RESERVED | | | |
| R/W-0h | | | | R-0h | | | |

**Table 13-20. CTRIPLFILCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | FILINIT | R-0/W1S | 0h | Low filter initialization.<br>0 No effect<br>1 Initialize all samples to the filter input value<br>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.<br>Reset type: SYSRSn |
| 8-4 | SAMPWIN | R/W | 0h | Low filter sample window size. Number of samples to monitor is SAMPWIN+1.<br>Reset type: SYSRSn |
| 3-0 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 13.8.2.18 CTRIPLFILCLKCTL Register (Offset = 17h) [Reset = 0000h]

CTRIPLFILCLKCTL is shown in Figure 13-24 and described in Table 13-21.

Return to the Summary Table.

CTRIPL Filter Clock Control Register

**Figure 13-24. CTRIPLFILCLKCTL Register**

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

**Table 13-21. CTRIPLFILCLKCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-10 | RESERVED | R | 0h | Reserved |
| 9-0 | CLKPRESCALE | R/W | 0h | Low filter sample clock prescale. Number of system clocks between samples is CLKPRESCALE+1.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 13.8.2.19 CTRIPHFILCTL Register (Offset = 18h) [Reset = 0000h]

CTRIPHFILCTL is shown in Figure 13-25 and described in Table 13-22.

Return to the Summary Table.

CTRIPH Filter Control Register

**Figure 13-25. CTRIPHFILCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| FILINIT | RESERVED | THRESH | | | | | SAMPWIN |
| R-0/W1S-0h | R-0h | R/W-0h | | | | | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SAMPWIN | | | | RESERVED | | | |
| R/W-0h | | | | R-0h | | | |

**Table 13-22. CTRIPHFILCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | FILINIT | R-0/W1S | 0h | High filter initialization.<br>0 No effect<br>1 Initialize all samples to the filter input value<br>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.<br>Reset type: SYSRSn |
| 8-4 | SAMPWIN | R/W | 0h | High filter sample window size. Number of samples to monitor is SAMPWIN+1.<br>Reset type: SYSRSn |
| 3-0 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 13.8.2.20 CTRIPHFILCLKCTL Register (Offset = 19h) [Reset = 0000h]

CTRIPHFILCLKCTL is shown in Figure 13-26 and described in Table 13-23.

Return to the Summary Table.

CTRIPH Filter Clock Control Register

**Figure 13-26. CTRIPHFILCLKCTL Register**

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

**Table 13-23. CTRIPHFILCLKCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-10 | RESERVED | R | 0h | Reserved |
| 9-0 | CLKPRESCALE | R/W | 0h | High filter sample clock prescale. Number of system clocks between samples is CLKPRESCALE+1.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 13.8.2.21 COMPLOCK Register (Offset = 1Ah) [Reset = 0000h]

COMPLOCK is shown in Figure 13-27 and described in Table 13-24.

Return to the Summary Table.

CMPSS Lock Register

**Figure 13-27. COMPLOCK Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | RESERVED | CTRIP | DACCTL | COMPHYSCTL | COMPCTL |
| R-0h | | | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h | R/WSonce-0h |

**Table 13-24. COMPLOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-5 | RESERVED | R | 0h | Reserved |
| 4 | RESERVED | R/WSonce | 0h | Reserved |
| 3 | CTRIP | R/WSonce | 0h | Lock write-access to the CTRIPxFILTCTL and CTRIPxFILCLKCTL* registers.<br>0 CTRIPxFILCTL and CTRIPxFILCLKCTL* registers are not locked. Write 0 to this bit has no effect.<br>1 CTRIPxFILCTL and CTRIPxFILCLKCTL* registers are locked. Only a system reset can clear this bit.<br>Reset type: SYSRSn |
| 2 | DACCTL | R/WSonce | 0h | Lock write-access to the COMPDAC*CTL* register(s).<br>0 COMPDAC*CTL* register(s) not locked. Write 0 to this bit has no effect.<br>1 COMPDAC*CTL* register(s) locked. Only a system reset can clear this bit.<br>Reset type: SYSRSn |
| 1 | COMPHYSCTL | R/WSonce | 0h | Lock write-access to the COMPHYSCTL register.<br>0 COMPHYSCTL register is not locked. Write 0 to this bit has no effect.<br>1 COMPHYSCTL register is locked. Only a system reset can clear this bit.<br>Reset type: SYSRSn |
| 0 | COMPCTL | R/WSonce | 0h | Lock write-access to the COMPCTL register.<br>0 COMPCTL register is not locked. Write 0 to this bit has no effect.<br>1 COMPCTL register is locked. Only a system reset can clear this bit.<br>Reset type: SYSRSn |

### *13.8.3 CMPSS Registers to Driverlib Functions*

**Table 13-25. CMPSS Registers to Driverlib Functions**

| File | Driverlib Function |
|---|---|
| **COMPCTL** | |
| cmpss.h | CMPSS_enableModule |
| cmpss.h | CMPSS_disableModule |
| cmpss.h | CMPSS_configHighComparator |
| cmpss.h | CMPSS_configLowComparator |
| cmpss.h | CMPSS_configOutputsHigh |
| cmpss.h | CMPSS_configOutputsLow |



Copyright © 2024 Texas Instruments Incorporated

**Table 13-25. CMPSS Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| **COMPHYSCTL** | |
| cmpss.h | CMPSS_setHysteresis |
| **COMPSTS** | |
| 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 |
| **COMPSTSCLR** | |
| 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 |
| **COMPDACCTL** | |
| cmpss.c | CMPSS_configRamp |
| cmpss.h | CMPSS_configDAC |
| cmpss.h | CMPSS_configureSyncSource |
| **DACHVALS** | |
| cmpss.h | CMPSS_setDACValueHigh |
| **DACHVALA** | |
| cmpss.h | CMPSS_getDACValueHigh |
| **RAMPMAXREFA** | |
| cmpss.h | CMPSS_getMaxRampValue |
| **RAMPMAXREFS** | |
| cmpss.c | CMPSS_configRamp |
| cmpss.h | CMPSS_setMaxRampValue |
| **RAMPDECVALA** | |
| cmpss.h | CMPSS_getRampDecValue |
| **RAMPDECVALS** | |
| cmpss.c | CMPSS_configRamp |
| cmpss.h | CMPSS_setRampDecValue |
| **RAMPSTS** | |
| - | |
| **DACLVALS** | |
| cmpss.h | CMPSS_setDACValueLow |
| **DACLVALA** | |
| cmpss.h | CMPSS_getDACValueLow |
| **RAMPDLYA** | |
| cmpss.h | CMPSS_getRampDelayValue |



Copyright © 2024 Texas Instruments Incorporated

**Table 13-25. CMPSS Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| **RAMPDLYS** | |
| cmpss.c | CMPSS_configRamp |
| cmpss.h | CMPSS_setRampDelayValue |
| **CTRIPLFILCTL** | |
| cmpss.c | CMPSS_configFilterLow |
| cmpss.h | CMPSS_initFilterLow |
| **CTRIPLFILCLKCTL** | |
| cmpss.c | CMPSS_configFilterLow |
| **CTRIPHFILCTL** | |
| cmpss.c | CMPSS_configFilterHigh |
| cmpss.h | CMPSS_initFilterHigh |
| **CTRIPHFILCLKCTL** | |
| cmpss.c | CMPSS_configFilterHigh |
| **COMPLOCK** | |
| - | |



Copyright © 2024 Texas Instruments Incorporated

Chapter 14

# Sigma Delta Filter Module (SDFM)

The sigma delta filter module (SDFM) is a four-channel digital filter designed specifically for current measurement and resolver position decoding in motor control applications. Each input channel can receive an independent delta-sigma (ΔΣ) modulator bit stream. The bit streams are processed by four individually-programmable digital decimation filters. The filter set includes a fast comparator (secondary filter) for immediate digital threshold comparisons for over-current and under-current monitoring, and zeros crossing detection.


**14.1 Introduction** ..... 1842
**14.2 Configuring Device Pins** ..... 1846
**14.3 Input Control Unit** ..... 1847
**14.4 Sinc Filter** ..... 1848
**14.5 Data (Primary) Filter Unit** ..... 1851
**14.6 Comparator (Secondary) Filter Unit** ..... 1854
**14.7 Theoretical SDFM Filter Output** ..... 1856
**14.8 Interrupt Unit** ..... 1858
**14.9 Register Descriptions** ..... 1860
**14.10 Software** ..... 1862
**14.11 SDFM Registers** ..... 1862





Copyright © 2024 Texas Instruments Incorporated

## 14.1 Introduction

Figure 14-1 shows the SDFM CPU Interface.

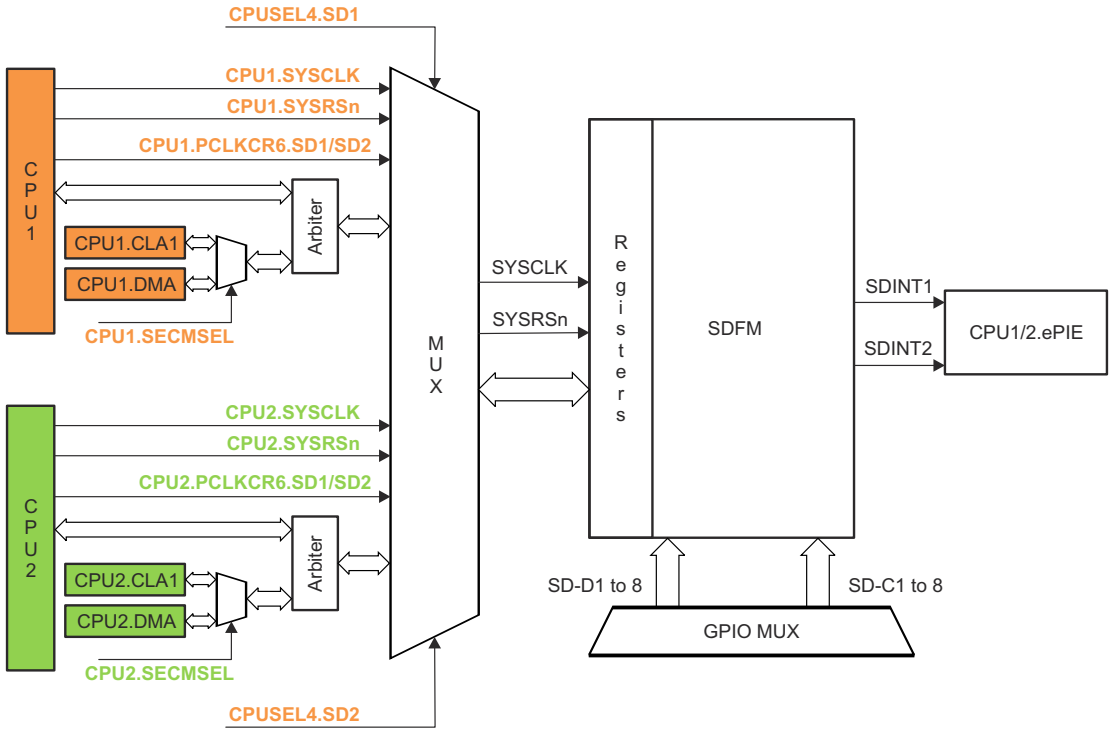



**Figure 14-1. Sigma Delta Filter Module (SDFM) CPU Interface**

### *14.1.1 SDFM Related Collateral*

**Foundational Materials**

- C2000 Academy - SDFM
- *How delta-sigma ADCs work, Part 1 Application Report*
- *How delta-sigma ADCs work, Part 2 Application Report*
- Nuts and Bolts of the Delta-Sigma Converter (Video)
- Sigma Delta Filter Module (SDFM) Training for C2000™ MCUs (Video)

**Getting Started Materials**

- *Achieving Better Signal Integrity With Isolated Delta-Sigma Modulators in Motor Drives Application Report*
  - Critical information for a hardware designer
- Using Sigma Delta Filter Module (SDFM) to Measure the Analog Input Signal
  - NOTE: This is a non-TI (third party) site.

**Expert Materials**

- *Clock Edge Delay Compensation With Isolated Modulators Digital Interface to MCUs Application Report*
- *Diagnosing Delta-Sigma Modulator Bitstream Using C2000™ Configurable Logic Block Application Report*
- *Isolated Current Shunt and Voltage Measurement Kit Application Report*
- Isolated, Shunt-Based Current Sensing Reference Design
- *Quick Response Control of PMSM Using Fast Current Loop Application Report*
- The case for isolated delta-sigma modulators: Is my system fast enough for short-circuit detection?
- Vienna Rectifier-Based Three Phase Power Factor Correction Reference Design Using C2000 MCU



Copyright © 2024 Texas Instruments Incorporated

### 14.1.2 Features

SDFM features include:

- Eight external pins per SDFM module
  - Four sigma-delta data input pins per SDFM module (SD-Dx, where x = 1 to 4)
  - Four sigma-delta clock input pins per SDFM module (SD-Cx, where x = 1 to 4)
- Different configurable modulator clock modes supported:
  - Mode 0: Modulator clock rate equals the modulator data rate.
  - Mode 1: Modulator clock rate running at half the modulator data rate.
  - Mode 2: Modulator data is Manchester-encoded. Modulator clock is not required.
  - Mode 3: Modulator clock rate is double that of the modulator data rate
- Four independent, configurable secondary filter (comparator) units per SDFM module:
  - Four different filter type selection (Sinc1/Sinc2/SincFast/Sinc3) options available
  - Ability to detect over-value condition, under-value condition, and Threshold-crossing conditions
  - OSR value for comparator filter unit (COSR) programmable from 1 to 32
- Four independent configurable primary filter (data filter) units per SDFM module:
  - Four different filter type selection (Sinc1/Sinc2/SincFast/Sinc3) options available
  - OSR value for data filter unit (DOSR) programmable from 1 to 256
  - Ability to enable or disable (or both) individual filter module
  - Ability to synchronize all four independent filters of an SDFM module by using the Master Filter Enable (MFE) bit or by using PWM signals
- Data filter output can be represented in either 16 bits or 32 bits.
- PWMs can be used to generate a modulator clock for sigma-delta modulators.



Copyright © 2024 Texas Instruments Incorporated

### 14.1.3 Block Diagram

Each SDFM module has four independent filter modules. These filter modules are identical and can be configured independently. Each individual filter module has the following units:

- Input control unit
- Primary filter (data filter) unit
- Secondary filter (comparator filter) unit with 4 independent comparators

Figure 14-2 shows the SDFM module block diagram. The SDFM port operation is configured and controlled by the registers listed in Section 14.11.

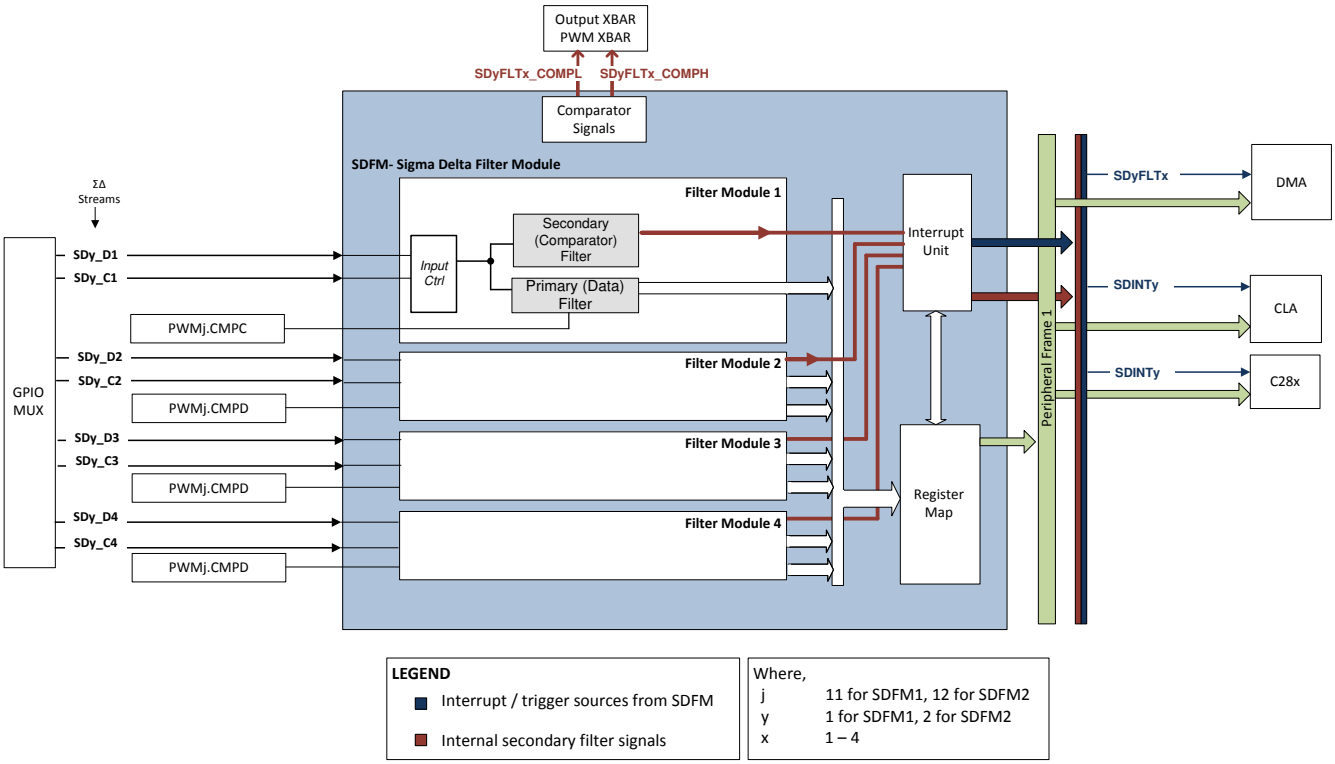



**Figure 14-2. Sigma Delta Filter Module (SDFM) Block Diagram**

**Note**

When using PWM11 or PWM12 for SDFM data filter synchronization, you must make sure that only one CMPC or CMPD event is generated per PWM timer period. Using PWM in up-count or down-count mode automatically makes sure that you get only one PWMC or PWMD event. But, if you want to use up-down count mode, then you need to make sure that only one CMPC or CMPD event per PWM cycle is generated; otherwise, the filter synchronizer corrupts the SDFM timing by providing two pulses per PWM cycle.

Each filter module shown in Figure 14-3 has a primary (data) filter and a secondary (comparator) filter pair that receives the same bit stream. Except for the input bit stream, both the primary and secondary filter are completely independent of each other. Each of these filter modules can be independently configured. So, in a SDFM module, there is a total of four primary filters and four secondary filters.



Copyright © 2024 Texas Instruments Incorporated

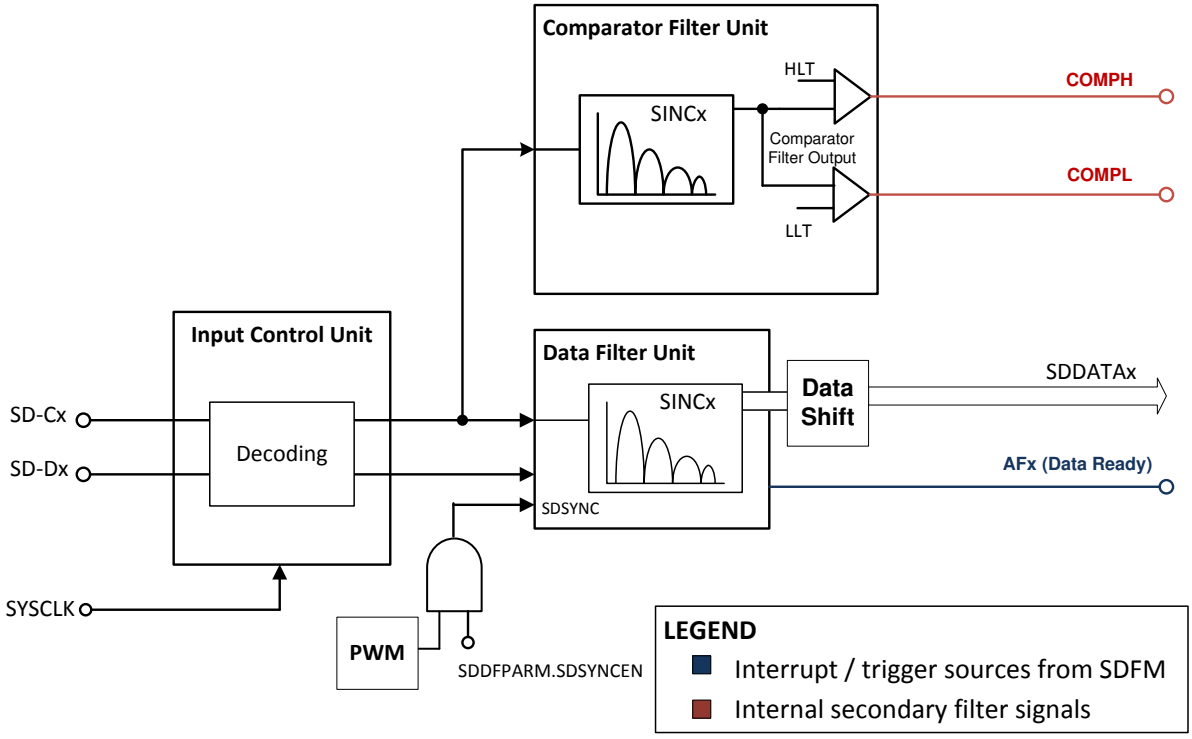



**Figure 14-3. Block Diagram of One Filter Module**



Copyright © 2024 Texas Instruments Incorporated

## 14.2 Configuring Device Pins

The GPIO mux registers must be configured to connect this peripheral to the device pins. To avoid glitches on the pins, the GPyGMUX bits must be configured first (while keeping the corresponding GPyMUX bits at the default of zero), followed by writing the GPyMUX register to the desired value.

For proper SDFM operation, use the following GPIO input qualification. Other GPIO qualifications are not supported.

- **Recommended option:** GPIO Input qualification = 3-sample GPIO Input Qualification. This option provides additional hardware protection against occasional random noise glitches. When GPIO Input qualification is a 3-sample window, make sure to check that the SDFM Electrical Data and Timing (Using 3-Sample GPIO Input Qualification) requirement is met and be aware of the following caution message and note.
- If GPIO Input qualification is ASYNC, make sure to check that the SDFM Electrical Data and Timing (Using ASYNC) requirement is met and be aware that this option is susceptible to random noise glitches.

**CAUTION**

The SDFM clock inputs (SDx_Cy pins) directly clock the SDFM module. Any glitches or ringing noise on these inputs can corrupt the SDFM module operation. Special precautions must be taken on these signals to make sure of a clean and noise-free signal that meets SDFM timing requirements. Precautions such as series termination for ringing due to any impedance mismatch of the clock driver and spacing of traces from other noisy signals are recommended.

**Note**

The SDFM module expects SD-Dx to change on the falling edge of SD-Cx and strobes for SD-Dx on the rising edge. But some SD-modulators in the market change SD-Dx on the rising edge and expect SDFM to strobe for data on the falling edge. In such cases, the GPIO inversion feature (GPxINV) is used on SD-Cx pin to change polarity and make it compatible with the SDFM.

The SDFM Qualified GPIO (3-sample) option provides protection against SDFM module corruption due to occasional random noise glitches on the SDx_Cy pin that can result in a false comparator trip and filter output.

The SDFM Qualified GPIO (3-sample) option does not provide protection against persistent violations of the above timing requirements. Timing violations results in data corruption proportional to the number of bits that violate the requirements.

See the *General-Purpose Input/Output (GPIO)* chapter for more details on GPIO mux and settings.



Copyright © 2024 Texas Instruments Incorporated

## 14.3 Input Control Unit

The input control unit receives sigma delta modulated data and a sigma delta modulated clock. The modulated data received is captured and passed on to the data filter unit and comparator unit. This unit can be configured to receive the modulated data in four different modes. Table 14-1 and Figure 14-4 show how SDCTLPARMx.MOD bits can be configured in these four different modes.

**Table 14-1. Modulator Clock Modes**

| Modulator Mode [MOD] | Description |
|---|---|
| 0 | The modulator clock is running with the modulator data rate. The modulator data is strobed at every rising edge of the modulator clock. |
| 1 | The modulator clock is running with half of the modulator data rate. The modulator data is strobed at every edge of the modulator clock. |
| 2 | The modulator clock is off and the modulator data is Manchester-encoded. |
| 3 | The modulator clock is running with double the modulator data rate. The modulator data is strobed at every other positive modulator clock edge. |

**Note**

To achieve the maximum value, the sigma-delta modulator has to be operated at absolute maximum positive or negative full scale, which is outside of the recommended full scale range of 80% of most sigma-delta modulators.

When MOD=2, data and modulated clock signals are encoded into modulated data as shown in Mode 2 of Figure 14-4. In this mode, the clock input SD-Cx pin can be left floating. The input control unit performs continuous automatic calibration to achieve optimum decoding performance.



Copyright © 2024 Texas Instruments Incorporated

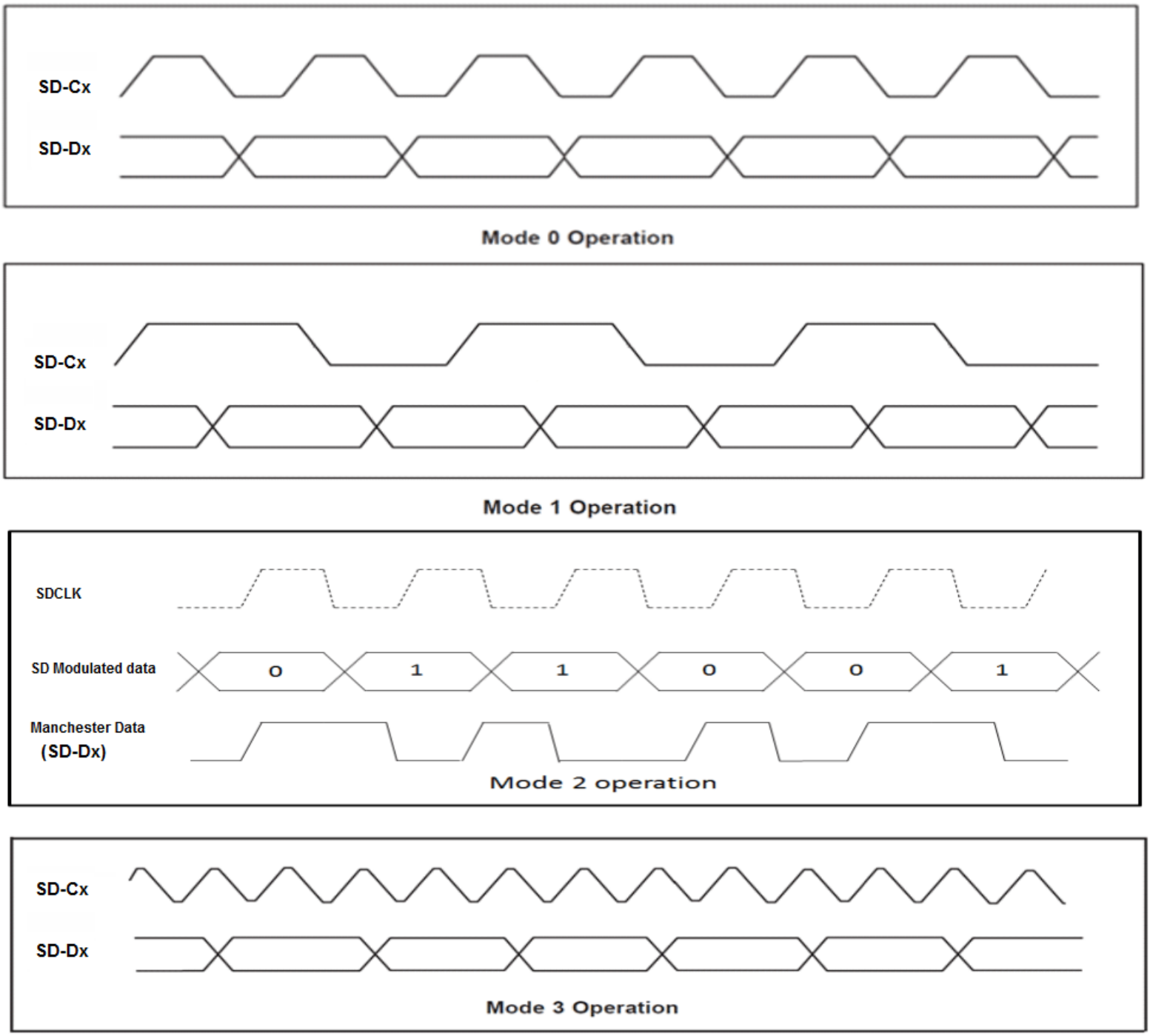



**Figure 14-4. Different Modulator Modes Supported**

## 14.4 Sinc Filter

Both the comparator filter and data filter available in SDFM have the Sinc$^N$ filter as the core. The Sinc$^N$ filter is essentially a low-pass filter that converts the input bit stream into digital data by digital filtering and decimation. This filtered digital data represents analog input given to the sigma delta modulator. Simplified Sinc$^N$ architecture



Copyright © 2024 Texas Instruments Incorporated

consists of cascaded integrators and differentiators separated by a down-sampler as shown in Figure 14-5. The Z-transfer function of the Sinc filter of order N is shown in Figure 14-6.

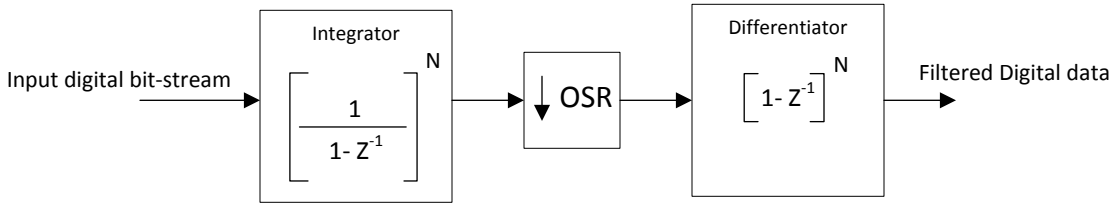



**Figure 14-5. Simplified Sinc Filter Architecture**

$$H(Z) = \left[ \frac{1 - Z^{-OSR}}{1 - Z^{-1}} \right]^{N}$$

N = Order of Sinc filter
OSR = Over Sampling Ratio

**Figure 14-6. Z-Transform of Sinc Filter of Order N**

Effective resolution of the Sinc filter (ENOB) depends upon filter type, OSR and sigma-delta modulator frequency. Typically, higher resolution or ENOB can be achieved by higher OSR for a given filter type; however, the tradeoff is increased filter delay. It is important to choose the right sigma delta modulator by studying the optimal speed versus resolution tradeoff. Refer to the corresponding sigma delta modulator data sheet to determine the effective resolution for a given Sinc filter configuration. Figure 14-7 shows the frequency response of different filter structures when OSR = 32 and when the sigma delta modulator frequency is 10MHz.

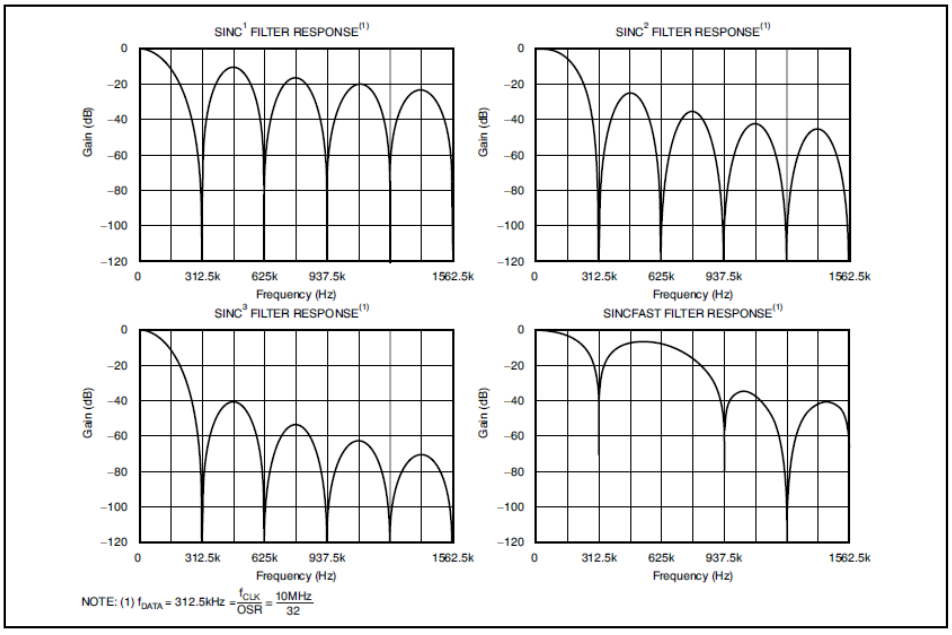



**Figure 14-7. Frequency Response of Different Sinc Filters**



Copyright © 2024 Texas Instruments Incorporated

The order of different sinc filter is shown in Table 14-2.

**Table 14-2. Order of Sinc Filter**

| Filter Type | Order of Sinc Filter |
|---|---|
| Sinc1 | 1 |
| Sinc2 | 2 |
| Sinc3 | 3 |
| SincFast | 3 |

### *14.4.1 Data Rate and Latency of the Sinc Filter*

The data rate of the sinc filter (filter throughput) represented in samples/sec is calculated by the following formula:

$$Data\ rate\ of\ Sinc\ filter = \frac{Modulator\ data\ rate}{OSR} \tag{17}$$

The latency of the sinc filter represented in secs is defined as the amount of time taken by a sinc filter type to deliver the correct filtered output upon initiation. For a given filter type, latency is calculated by the following formula:

$$Latency\ of\ Sinc\ filter = \frac{Order\ of\ Sinc\ filter}{Data\ rate\ of\ Sinc\ filter} \tag{18}$$

***Example configuration:***

| | |
|---|---|
| Sinc filter type | = sinc3 |
| Modulator data rate | = 10MHz |
| OSR | = 256 |
| Data rate of Sinc Filter | = 10MHz/256 = 39.1K samples/second |
| Sinc filter latency | = 76.8µs |
| Sinc filter type | = sinc2 |
| Modulator data rate | = 10MHz |
| OSR | = 256 |
| Data rate of Sinc Filter | = 10MHz/256 = 39.1K samples/second |
| Sinc filter latency | =51.2µs |



Copyright © 2024 Texas Instruments Incorporated

## 14.5 Data (Primary) Filter Unit

The data filter is a configurable Sinc filter which supports the following filter types: Sinc1, Sinc2, Sinc3, and SincFast. The data filter OSR (DOSR) settings can be configured from 1 to 256 and is independent of the comparator filter. Effective resolution of the data filter (ENOB) depends upon Data filter type, DOSR, and sigma-delta modulator frequency. By default, the data filter is disabled and setting of SDDFPARMx.FEN = 1 enables the data filter. The data filter output is represented in 25-bit signed integer in two's complement format. This filter unit translates a low input signal as '-1' and a high input signal as '1'. The resulting calculation gives both positive and negative values for the output of the data filter. Table 14-3 shows the different full scale values that the data filter can store using different OSRs.

See Section 14.4.1 to understand how to calculate data rate and latency of data filter.

**Table 14-3. Peak Data Values for Different DOSR/Filter Combinations**

| DOSR | Sinc1 | Sinc2 | Sinc3 | SincFast |
|---|---|---|---|---|
| x | x | $x^2$ | $x^3$ | $2x^2$ |
| 4 | –4 to 4 | –16 to 16 | –64 to 64 | –32 to 32 |
| 8 | –8 to 8 | –64 to 64 | –512 to 512 | –128 to 128 |
| 16 | –16 to 16 | –256 to 256 | –4096 to 4096 | –512 to 512 |
| 32 | –32 to 32 | –1024 to 1024 | –32,768 to 32,768 | –2048 to 2048 |
| 64 | –64 to 64 | –4096 to 4096 | –262,144 to 262,144 | –8192 to 8192 |
| 128 | –128 to 128 | –16,384 to 16,384 | –2,097,152 to 2,097,152 | –32,768 to 32,768 |
| 256 | –256 to 256 | –65,536 to 65,536 | –16,777,216 to 16,777,216 | –131,072 to 131,072 |



Copyright © 2024 Texas Instruments Incorporated

### 14.5.1 32-bit or 16-bit Data Filter Output Representation

The data filter output can be represented in either 32-bit or 16-bit format.

**32-bit data filter representation:**

- When SDDPARMx.DR = 1, data filter output is represented in 32-bit format. Writes to shift control bits do not have any bearing on the output of the data filter in this configuration.

**16-bit data filter representation:**

- By default, data filter output is represented in 16-bit format
- When SDDPARMx.DR = 0, data filter output is represented in 16-bit format. But it is the responsibility of the user to configure the corresponding shift control bits in the SDDPARMx register to control which 16-bit part of the 32-bit word is sent to the register map.

  For example, for the data filter configuration below:
  - Filter type = Sinc3
  - OSR = 128
  - SDDPARMx.DR = 0

  The data filter with a 25-bit signed output value can be in the range of –2,097,152 to 2,097,152. But, 16-bit signed output supports values only from –32,768 to 32,767. Therefore, it is required to configure shift control bits (SDDPARMx.SH) to 7 to represent the data filter output correctly in 16-bit format. Table 14-4 shows the configuration settings of shift control bits for different OSR and filter types.

**Table 14-4. Shift Control Bit Configuration Settings**

| OSR | Sinc1 | Sinc2 | SincFast | Sinc3 |
|---|---|---|---|---|
| 1 to 31 | 0 | 0 | 0 | 0 |
| 32 to 40 | 0 | 0 | 0 | 1 |
| 41 to 50 | 0 | 0 | 0 | 2 |
| 51 to 63 | 0 | 0 | 0 | 3 |
| 64 to 80 | 0 | 0 | 0 | 4 |
| 81 to 101 | 0 | 0 | 0 | 5 |
| 102 to 127 | 0 | 0 | 0 | 6 |
| 128 to 161 | 0 | 0 | 1 | 7 |
| 162 to 181 | 0 | 0 | 1 | 8 |
| 182 to 203 | 0 | 1 | 2 | 8 |
| 204 to 255 | 0 | 1 | 2 | 9 |
| 256 | 0 | 2 | 3 | 9 |

**CAUTION**

Configuring shift control bits incorrectly results in getting an incorrect 16-bit data filter output.



Copyright © 2024 Texas Instruments Incorporated

### 14.5.2 SDSYNC Event

Primary (data) filters can be synchronized with respect to the PWM event (called SDSYNC event). The SDSYNC signal from the PWM module is used to reset the DOSR counter. This feature is by default disabled and can be enabled by setting SDDFPARMx.SDSYNCEN = 1. Figure 14-8 shows how the PWM signals are connected to the sigma delta module. In this device, PWM11 can be used to reset SDFM1 filter modules and PWM12 can be used to reset SDFM2 modules.

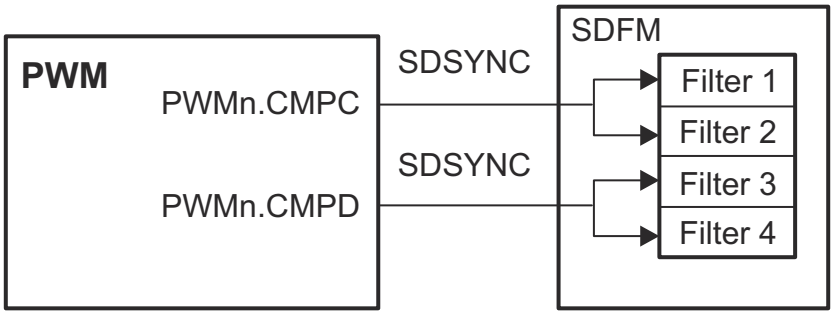



**Figure 14-8. SDSYNC Event**

Because of the inherent architecture of the Sinc filter (Sinc1, Sinc2, Sinc3, SincFast), the first few samples, depending upon filter type, are incorrect. Table 14-5 shows the number of incorrect samples on the following conditions:

- When Sinc filter is enabled and configured for first time.
- When Sinc filter is disabled and re-enabled or reconfigured in the middle of operation.
- When data filter receives SDSYNC event from PWM.

**Table 14-5. Number of Incorrect Samples Tabulated**

| Filter Type | Number of Incorrect Samples After the Filter is Enabled and Configured |
|---|---|
| Sinc1 | No incorrect sample. |
| Sinc2 | The first sample of the Sinc2 filter is incorrect. |
| SincFast | The first two samples of the SincFast filter are incorrect. |
| Sinc3 | The first two samples of the Sinc3 filter are incorrect. |

**CAUTION**

SDFM comparator interrupts can be enabled only after providing sufficient settling time to make sure the comparator filter does not trip on these incorrect samples. Therefore, SDFM comparator interrupts (IELx and IEHx) can be enabled only after a sufficient delay is provided after the comparator filter is configured. This sufficient delay is calculated by adding the latency of the comparator filter and 5 SD-Cx clock cycles.



Copyright © 2024 Texas Instruments Incorporated

## 14.6 Comparator (Secondary) Filter Unit

Most control systems require protection of the system by tripping the PWM in case the current or voltage goes out of bounds. The primary purpose of the secondary (comparator) filter is to allow the user to monitor input conditions with a fast settling time. This allows the user to trip PWMs to protect the system from potential damage.

**Note**

The secondary (comparator) filter cannot be synchronized with respect to the PWM event (SDSYNC event).

The comparator filter is a configurable Sinc filter that supports the following filter types: Sinc1, Sinc2, Sinc3, and SincFast. The comparator OSR (COSR) settings can be configured from 1 to 32 and is independent of the data filter. Effective resolution of the comparator filter (ENOB) depends upon the comparator filter type, COSR, and sigma-delta modulator frequency. By default, the comparator filter is enabled. The comparator filter output is represented in 16-bit unsigned format. This filter unit translates a low input signal as 0 and a high input signal as 1. The resulting calculations give only positive values for the output of the comparator filter. Table 14-6 shows the different full-scale values that the comparator filter can store using different OSRs.

**Table 14-6. Peak Data Values for Different OSR/Filter Combinations**

| OSR | Sinc1 | Sinc2 | Sinc3 | SincFast |
|---|---|---|---|---|
| x | 0 to x | 0 to x2 | 0 to x3 | 0 to 2x2 |
| 4 | 0 to 4 | 0 to 16 | 0 to 64 | 0 to 32 |
| 8 | 0 to 8 | 0 to 64 | 0 to 512 | 0 to 128 |
| 16 | 0 to 16 | 0 to 256 | 0 to 4096 | 0 to 512 |
| 32 | 0 to 32 | 0 to 1024 | 0 to 32,768 | 0 to 2048 |

See Section 14.4.1 to understand how to calculate data rate and latency of comparator filter.

The output of the comparator filter is not memory-mapped; instead the output is connected to digital comparators explained below.

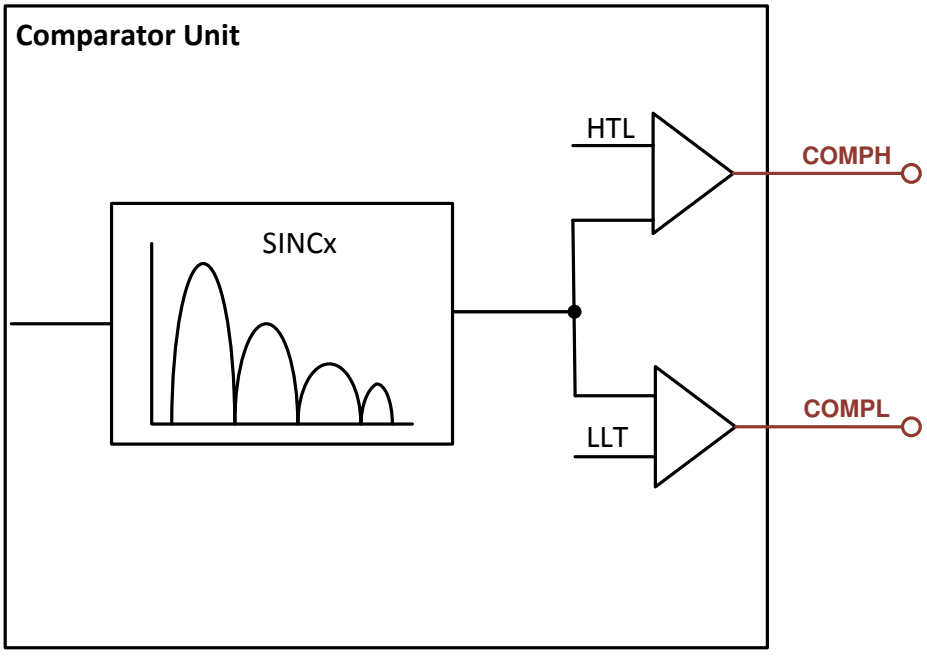



**Figure 14-9. Comparator Unit Structure**



Copyright © 2024 Texas Instruments Incorporated

### 14.6.1 Higher Threshold (HLT) Comparator

- High threshold comparator can be used to detect over-value condition.
- When comparator data > = higher threshold register, a high threshold event is generated.
- Higher threshold comparator events can be configured to trigger following events: CPU interrupt, CLA task, PWM trip.
- This device has one high threshold comparator:
  - **Higher Threshold (HLT) Comparator**: When SDCTL.MIE = 1 and SDCPARMx.IEH = 1, the high threshold event triggers an SDFM interrupt (SDINT) and sets the SDIFLG.IEHx flag showing the over-value condition is triggered. This SDIFLG.IEHx flag can be reset if the corresponding bit in the SDIFLGCLR register is set and if the interrupt source is no longer active.

### 14.6.2 Lower Threshold (LLT) Comparator

- The low threshold comparator can be used to detect under-value condition.
- When comparator data < = Lower Threshold register, a low threshold event is generated.
- Lower threshold comparator events can be configured to trigger following events: CPU interrupt, CLA task, PWM trip.
- This device has one low threshold comparator..
  - **Lower Threshold (LLT) Comparator** : When SDCTL.MIE = 1 and SDCPARMx.IEL = 1, the low threshold event triggers an SDFM interrupt (SDINT) and sets the SDIFLG.IELx flag showing an under-value condition is triggered. This SDIFLG.IELx flag can be reset, if the corresponding bit in the SDIFLGCLR register is set and if the interrupt source is no longer active.



Copyright © 2024 Texas Instruments Incorporated

## 14.7 Theoretical SDFM Filter Output

The following equations can be used to derive a theoretical filter output of an SDFM filter output for both a comparator filter and a data filter.

$$\text{Density of ones in bitstream} = \frac{\text{Input Voltage} + \text{Vclipping}}{2 \times \text{Vclipping}} \tag{19}$$

Where:

- Vclipping = maximum differential voltage input range of modulator
- Input voltage = Differential input voltage applied to the modulator

$$\text{Comparator Filter Output (Theoretical)} = \text{Density of ones in bitstream} \times \text{Maximum Filter Output (FilterType, COSR)} \tag{20}$$

$$\text{FilterOutput} = \left\{\frac{\text{absolute(Input voltage)}}{\text{Vclipping}}\right\} \times \text{Maximum Filter Output (FilterType, DOSR)} \tag{21}$$

$$\text{Data Filter Output_32bit(Theoretical)} = \begin{cases} \text{FilterOutput} & \text{if Input Voltage is +ve voltage} \\ \text{2's complement of FilterOutput} & \text{if input voltage is -ve voltage} \end{cases} \tag{22}$$

$$\text{Data Filter Output_16bit(Theoretical)} = \text{Data Filter Output_32bit(Theoretical)} >> \text{Shift value(FilterType, OSR)} \tag{23}$$



Copyright © 2024 Texas Instruments Incorporated

For example, when using the AMC1306x25 modulator:

| | | |
|---|---|---|
| AMC1306x25 | Vclipping =<br>Input voltage (AINP - AINN) = | 320mV<br>100mV |
| SDFM filter settings | Filter type =<br>Comparator OSR (COSR) =<br>Data filter OSR (DOSR) = | 3<br>32<br>100 |

| | | |
|---|---|---|
| Density of 1s in bit stream | Using Equation 19 | 0.65625 |
| Comparator filter output<br>Filter type = Sinc3<br>COSR = 32 | Using Equation 20 | 21504 |
| Data filter output (32-bit)<br>Filter type = Sinc3<br>DOSR = 100 | Using Equation 21 and Equation 22 | 312500 |
| Data filter output (32-bit)<br>Filter type = Sinc3<br>DOSR = 100<br>(Right shift by 5) | Using Equation 23 | 9765 |



Copyright © 2024 Texas Instruments Incorporated

## 14.8 Interrupt Unit

Each SDFM can generate one CPU interrupt (SDINT).

### *14.8.1 SDFM (SDINT) Interrupt Sources*

Figure 14-10 shows the structure of the interrupt unit. Each SDFM module can generate a CPU interrupt. An SDFM interrupt can be triggered by any of these 16 events.

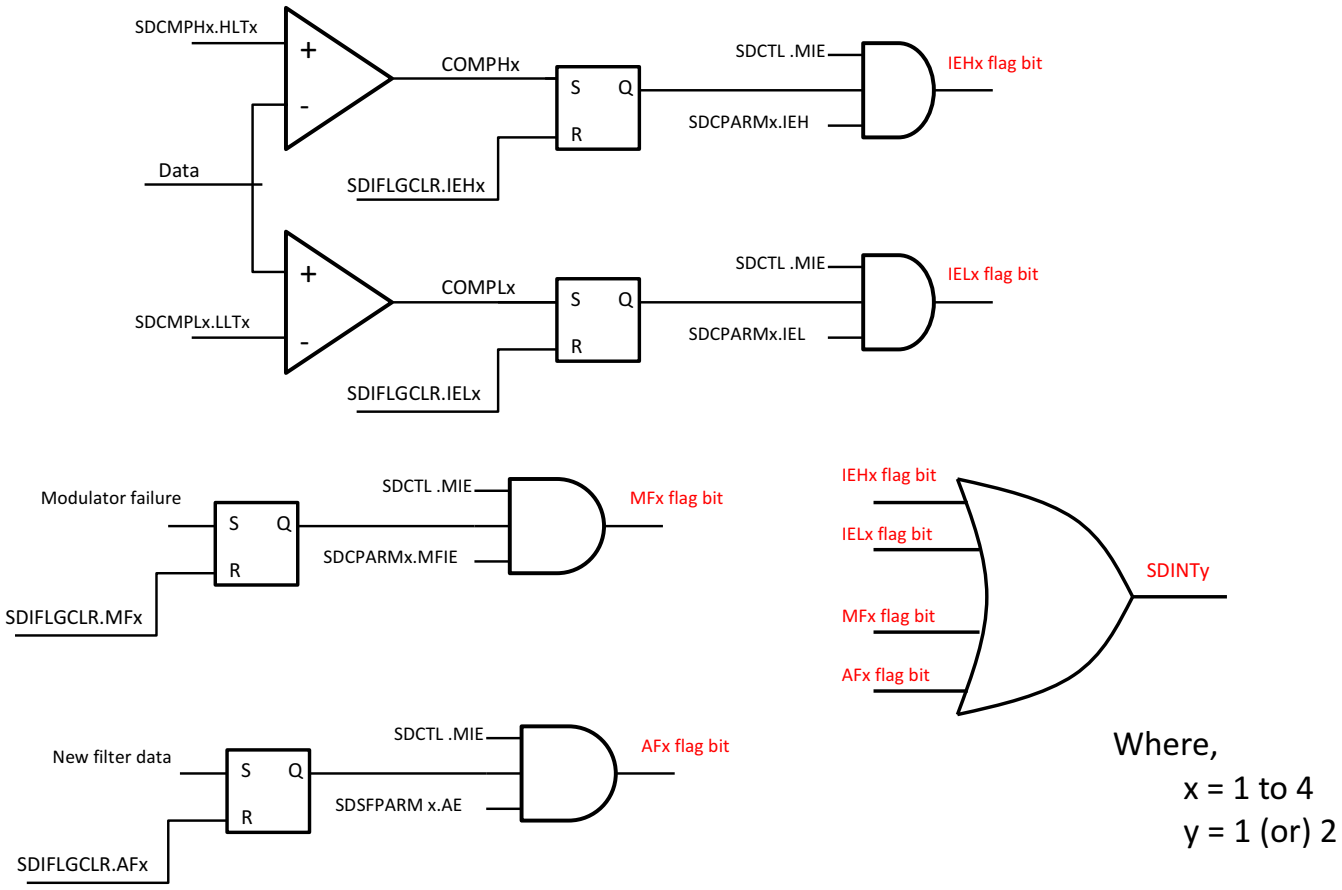



**Figure 14-10. SDFM Error (SD_ERR) Interrupt Sources**



Copyright © 2024 Texas Instruments Incorporated

**1. Comparator Lower Threshold events (COMPLx)**

COMPL events from any of the four comparator filter module can trigger CPU interrupt. This event can be configured to trigger SDINT interrupt only if below configurations are made:

- Enable Main interrupt enable (SDCTL.MIE = 1)
- Enable comparator filter lower threshold interrupt (SDCPARMx.IEL = 1)

On a COMPL event, SDIFLG.IELx flag bit is set. This flag bit can only be reset if the corresponding bit in SDIFLGCLR register is set and if the interrupt source is no longer active.

**2. Comparator High Threshold events (COMPHx)**

COMPH events from any of the four comparator filter module can trigger CPU interrupt. This event can be configured to trigger SDINT interrupt only if below configurations are made:

- Enable Main interrupt enable (SDCTL.MIE = 1)
- Enable comparator filter lower threshold interrupt (SDCPARMx.IEH = 1)

On a COMPH event, SDIFLG.IELx flag bit is set. This flag bit can only be reset if the corresponding bit in SDIFLGCLR register is set and if the interrupt source is no longer active.

**3. Modulator Failure (MFx) event**

Modulator failures (MFx) are generated when SD-Cx goes missing. The modulator clock is considered missing if SD-Cx does not toggle for 64-SYSCLKs. MFx events from any of the four filter modules can trigger CPU interrupt. This event can be configured to trigger SDINT interrupt only if below configurations are made:

- Enable Main Interrupt Enable (SDCTL.MIE = 1)
- Enable modulator clock failure interrupt source (SDCPARMx.MFIE = 1)

On a MFx event, SDIFLG.MFx flag bit is set. This flag bit can only be reset if the corresponding bit in SDIFLGCLR register is set and if the interrupt source is no longer active.

**4. New Filter Data Ready (AFx) event**

When the primary filter is ready with a new filter data, the AFx event is generated. AFx events from any of the four primary filter modules can be configured to trigger a CPU interrupt. This event can be configured to trigger SDINT interrupt only if below configurations are made:

- Enable Main interrupt enable (SDCTL.MIE = 1)
- Enable data filter acknowledge (SDDFPARMx.AE = 1)

On an AFx event, the SDIFLG.AFx flag bit is set. This flag bit can only be reset if the corresponding bit in SDIFLGCLR register is set and if the interrupt source is no longer active.



Copyright © 2024 Texas Instruments Incorporated

## 14.9 Register Descriptions

The register descriptions are shown in the following table and subsections.

**Table 14-7. General Registers**

| Name | SDFM1 address | SDFM2 address | Size (x16) | Description | EALLOW? |
|---|---|---|---|---|---|
| SDIFLG | 0x5E00 | 0x5E80 | 2 | Interrupt Flag Register | |
| SDIFLGCLR | 0x5E02 | 0x5E82 | 2 | Interrupt Flag Clear Register | |
| SDCTL | 0x5E04 | 0x5E84 | 1 | SD Control Register | YES |
| SDMFILEN | 0x5E06 | 0x5E86 | 1 | SD Master Filter Enable | YES |
| Reserved | 0x5E07 | 0x5E87 | 1 | Reserved | |

**Table 14-8. Filter 1 Registers**

| Name | SDFM1 address | SDFM2 address | Size (x16) | Description | EALLOW? |
|---|---|---|---|---|---|
| SDCTLPARM1 | 0x5E10 | 0x5E90 | 1 | Control Parameter Register for Ch1 | YES |
| SDDFPARM1 | 0x5E11 | 0x5E91 | 1 | Data Filter Parameter Register for Ch1 | YES |
| SDDPARM1 | 0x5E12 | 0x5E92 | 1 | Integer Parameter Register for Ch1 | YES |
| SDCMPH1 | 0x5E13 | 0x5E93 | 1 | High-level Threshold Register for Ch1 | YES |
| SDCMPL1 | 0x5E14 | 0x5E94 | 1 | Low-level Threshold Register for Ch1 | YES |
| SDCPARM1 | 0x5E15 | 0x5E95 | 1 | Comparator Parameter Register for Ch1 | YES |
| SDDATA1 | 0x5E16 | 0x5E96 | 2 | Filter Data Register (16- or 32-bit) for Ch1 | |

**Table 14-9. Filter 2 Registers**

| Name | SDFM1 address | SDFM2 address | Size (x16) | Description | EALLOW? |
|---|---|---|---|---|---|
| SDCTLPARM2 | 0x5E20 | 0x5EA0 | 1 | Control Parameter Register for Ch2 | YES |
| SDDFPARM2 | 0x5E21 | 0x5EA1 | 1 | Data Filter Parameter Register for Ch2 | YES |
| SDDPARM2 | 0x5E22 | 0x5EA2 | 1 | Integer Parameter Register for Ch2 | YES |
| SDCMPH2 | 0x5E23 | 0x5EA3 | 1 | High-level Threshold Register for Ch2 | YES |
| SDCMPL2 | 0x5E24 | 0x5EA4 | 1 | Low-level Threshold Register for Ch2 | YES |
| SDCPARM2 | 0x5E25 | 0x5EA5 | 1 | Comparator Parameter Register for Ch2 | YES |
| SDDATA2 | 0x5E26 | 0x5EA6 | 2 | Filter Data Register (16- or 32-bit) for Ch2 | |



Copyright © 2024 Texas Instruments Incorporated

**Table 14-10. Filter 3 Registers**

| Name | SDFM1 address | SDFM2 address | Size (x16) | Description | EALLOW? |
|---|---|---|---|---|---|
| SDCTLPARM3 | 0x5E30 | 0x5EB0 | 1 | Control Parameter Register for Ch3 | YES |
| SDDFPARM3 | 0x5E31 | 0x5EB1 | 1 | Data Filter Parameter Register for Ch3 | YES |
| SDDPARM3 | 0x5E32 | 0x5EB2 | 1 | Integer Parameter Register for Ch3 | YES |
| SDCMPH3 | 0x5E33 | 0x5EB3 | 1 | High-level Threshold Register for Ch3 | YES |
| SDCMPL3 | 0x5E34 | 0x5EB4 | 1 | Low-level Threshold Register for Ch3 | YES |
| SDCPARM3 | 0x5E35 | 0x5EB5 | 1 | Comparator Parameter Register for Ch3 | YES |
| SDDATA3 | 0x5E36 | 0x5EB6 | 2 | Filter Data Register (16- or 32-bit) for Ch3 | |

**Table 14-11. Filter 4 Registers**

| Name | SDFM1 address | SDFM2 address | Size (x16) | Description | EALLOW? |
|---|---|---|---|---|---|
| SDCTLPARM4 | 0x5E40 | 0x5EC0 | 1 | Control Parameter Register for Ch4 | YES |
| SDDFPARM4 | 0x5E41 | 0x5EC1 | 1 | Data Filter Parameter Register for Ch4 | YES |
| SDDPARM4 | 0x5E42 | 0x5EC2 | 1 | Integer Parameter Register for Ch4 | YES |
| SDCMPH4 | 0x5E43 | 0x5EC3 | 1 | High-level Threshold Register for Ch4 | YES |
| SDCMPL4 | 0x5E44 | 0x5EC4 | 1 | Low-level Threshold Register for Ch4 | YES |
| SDCPARM4 | 0x5E45 | 0x5EC5 | 1 | Comparator Parameter Register for Ch4 | YES |
| SDDATA4 | 0x5E46 | 0x5EC6 | 2 | Filter Data Register (16- or 32-bit) for Ch4 | |



Copyright © 2024 Texas Instruments Incorporated

## 14.10 Software

### 14.10.1 SDFM Examples

NOTE: These examples are located in the C2000Ware installation at the following location:
C2000Ware_VERSION#/driverlib/DEVICE_GPN/examples/CORE_IF_MULTICORE/sdfm

Cloud access to these examples is available at the following link: dev.ti.com C2000Ware Examples.

## 14.11 SDFM Registers

This section describes the Sigma Delta Filter Module registers.

### 14.11.1 SDFM Base Addresses

**Table 14-12. SDFM Base Address Table**

| Device Registers | Register Name | Start Address | End Address |
|---|---|---|---|
| Sdfm1Regs | SDFM_REGS | 0x0000_5E00 | 0x0000_5E7F |
| Sdfm2Regs | SDFM_REGS | 0x0000_5E80 | 0x0000_5EFF |



Copyright © 2024 Texas Instruments Incorporated

### 14.11.2 SDFM_REGS Registers

Table 14-13 lists the memory-mapped registers for the SDFM_REGS registers. All register offset addresses not listed in Table 14-13 should be considered as reserved locations and the register contents should not be modified.

**Table 14-13. SDFM_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | SDIFLG | Interrupt Flag Register | | Go |
| 2h | SDIFLGCLR | Interrupt Flag Clear Register | | Go |
| 4h | SDCTL | SD Control Register | EALLOW | Go |
| 6h | SDMFILEN | SD Master Filter Enable | EALLOW | Go |
| 10h | SDCTLPARM1 | Control Parameter Register for Ch1 | EALLOW | Go |
| 11h | SDDFPARM1 | Data Filter Parameter Register for Ch1 | EALLOW | Go |
| 12h | SDDPARM1 | Integer Parameter Register for Ch1 | EALLOW | Go |
| 13h | SDCMPH1 | High-level Threshold Register for Ch1 | EALLOW | Go |
| 14h | SDCMPL1 | Low-level Threshold Register for Ch1 | EALLOW | Go |
| 15h | SDCPARM1 | Comparator Parameter Register for Ch1 | EALLOW | Go |
| 16h | SDDATA1 | Filter Data Register (16 or 32bit) for Ch1 | | Go |
| 20h | SDCTLPARM2 | Control Parameter Register for Ch2 | EALLOW | Go |
| 21h | SDDFPARM2 | Data Filter Parameter Register for Ch2 | EALLOW | Go |
| 22h | SDDPARM2 | Integer Parameter Register for Ch2 | EALLOW | Go |
| 23h | SDCMPH2 | High-level Threshold Register for Ch2 | EALLOW | Go |
| 24h | SDCMPL2 | Low-level Threshold Register for Ch2 | EALLOW | Go |
| 25h | SDCPARM2 | Comparator Parameter Register for Ch2 | EALLOW | Go |
| 26h | SDDATA2 | Filter Data Register (16 or 32bit) for Ch2 | | Go |
| 30h | SDCTLPARM3 | Control Parameter Register for Ch3 | EALLOW | Go |
| 31h | SDDFPARM3 | Data Filter Parameter Register for Ch3 | EALLOW | Go |
| 32h | SDDPARM3 | Integer Parameter Register for Ch3 | EALLOW | Go |
| 33h | SDCMPH3 | High-level Threshold Register for Ch3 | EALLOW | Go |
| 34h | SDCMPL3 | Low-level Threshold Register for Ch3 | EALLOW | Go |
| 35h | SDCPARM3 | Comparator Parameter Register for Ch3 | EALLOW | Go |
| 36h | SDDATA3 | Filter Data Register (16 or 32bit) for Ch3 | | Go |
| 40h | SDCTLPARM4 | Control Parameter Register for Ch4 | EALLOW | Go |
| 41h | SDDFPARM4 | Data Filter Parameter Register for Ch4 | EALLOW | Go |
| 42h | SDDPARM4 | Integer Parameter Register for Ch4 | EALLOW | Go |
| 43h | SDCMPH4 | High-level Threshold Register for Ch4 | EALLOW | Go |
| 44h | SDCMPL4 | Low-level Threshold Register for Ch4 | EALLOW | Go |
| 45h | SDCPARM4 | Comparator Parameter Register for Ch4 | EALLOW | Go |
| 46h | SDDATA4 | Filter Data Register (16 or 32bit) for Ch4 | | Go |

Complex bit access types are encoded to fit into small table cells. Table 14-14 shows the codes that are used for access types in this section.

**Table 14-14. SDFM_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| R-0 | R<br>-0 | Read<br>Returns 0s |



Copyright © 2024 Texas Instruments Incorporated

**Table 14-14. SDFM_REGS Access Type Codes (continued)**

| Access Type | Code | Description |
|---|---|---|
| Write Type | | |
| W | W | Write |
| W1S | W<br>1S | Write<br>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. |



Copyright © 2024 Texas Instruments Incorporated

### 14.11.2.1 SDIFLG Register (Offset = 0h) [Reset = 00000000h]

SDIFLG is shown in Figure 14-11 and described in Table 14-15.

Return to the Summary Table.

Interrupt Flag Register

**Figure 14-11. SDIFLG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MIF | 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 |
| AF4 | AF3 | AF2 | AF1 | MF4 | MF3 | MF2 | MF1 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| IFL4 | IFH4 | IFL3 | IFH3 | IFL2 | IFH2 | IFL1 | IFH1 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 14-15. SDIFLG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | MIF | R | 0h | Set whenever any interrupt (ACK1-4, MF1-4,IFL1-4,IFH1-4) is active<br>Reset type: SYSRSn |
| 30-16 | RESERVED | R-0 | 0h | Reserved |
| 15 | AF4 | R | 0h | 0: No new data available from Filter 4<br>1: New data available from Filter 4<br>Reset type: SYSRSn |
| 14 | AF3 | R | 0h | 0: No new data available from Filter 3<br>1: New data available from Filter 3<br>Reset type: SYSRSn |
| 13 | AF2 | R | 0h | 0: No new data available from Filter 2<br>1: New data available from Filter 2<br>Reset type: SYSRSn |
| 12 | AF1 | R | 0h | 0: No new data available from Filter 1<br>1: New data available from Filter 1<br>Reset type: SYSRSn |
| 11 | MF4 | R | 0h | 0: Modulator is operating normally for Filter 4<br>1: Modulator failure for Filter 4<br>Reset type: SYSRSn |
| 10 | MF3 | R | 0h | 0: Modulator is operating normally for Filter 3<br>1: Modulator failure for Filter 3<br>Reset type: SYSRSn |
| 9 | MF2 | R | 0h | 0: Modulator is operating normally for Filter 2<br>1: Modulator failure for Filter 2<br>Reset type: SYSRSn |
| 8 | MF1 | R | 0h | 0: Modulator is operating normally for Filter 1<br>1: Modulator failure for Filter 1<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 14-15. SDIFLG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 7 | IFL4 | R | 0h | 0: Comparator Filter 4 output is above the low limit threshold<br>1: Comparator Filter 4 output is equal to or below the low level threshold, if enabled<br>Reset type: SYSRSn |
| 6 | IFH4 | R | 0h | 0: Comparator Filter 4 output is below the high limit threshold<br>1: Comparator Filter 4 output is equal to or above the high level threshold, if enabled<br>Reset type: SYSRSn |
| 5 | IFL3 | R | 0h | 0: Comparator Filter 3 output is above the low limit threshold<br>1: Comparator Filter 3 output is equal to or below the low level threshold, if enabled<br>Reset type: SYSRSn |
| 4 | IFH3 | R | 0h | 0: Comparator Filter 3 output is below the high limit threshold<br>1: Comparator Filter 3 output is equal to or above the high level threshold, if enabled<br>Reset type: SYSRSn |
| 3 | IFL2 | R | 0h | 0: Comparator Filter 2 output is above the low limit threshold<br>1: Comparator Filter 2 output is equal to or below the low level threshold, if enabled<br>Reset type: SYSRSn |
| 2 | IFH2 | R | 0h | 0: Comparator Filter 2 output is below the high limit threshold<br>1: Comparator Filter 2 output is equal to or above the high level threshold, if enabled<br>Reset type: SYSRSn |
| 1 | IFL1 | R | 0h | 0: Comparator Filter 1 output is above the low limit threshold<br>1: Comparator Filter 1 output is equal to or below the low level threshold, if enabled<br>Reset type: SYSRSn |
| 0 | IFH1 | R | 0h | 0: Comparator Filter 1 output is below the high limit threshold<br>1: Comparator Filter 1 output is equal to or above the high level threshold, if enabled<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 14.11.2.2 SDIFLGCLR Register (Offset = 2h) [Reset = 00000000h]

SDIFLGCLR is shown in Figure 14-12 and described in Table 14-16.

Return to the Summary Table.

Interrupt Flag Clear Register

**Figure 14-12. SDIFLGCLR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MIF | RESERVED | | | | | | |
| R-0/W1S-0h | R-0-0h | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| AF4 | AF3 | AF2 | AF1 | MF4 | MF3 | MF2 | MF1 |
| 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 |
| IFL4 | IFH4 | IFL3 | IFH3 | IFL2 | IFH2 | IFL1 | IFH1 |
| 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 14-16. SDIFLGCLR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | MIF | R-0/W1S | 0h | Flag-clear bit for SDFM Master Interrupt flag. Write 1 to clear MIF. Writes of '0' are ignored.<br>Note: If the MIF flag is cleared and other Interrupts are still pending, MIF will again be set to 1 on the following SysClk cycle, and the INT output will be reasserted (pulsed low)<br>Reset type: SYSRSn |
| 30-16 | RESERVED | R-0 | 0h | Reserved |
| 15 | AF4 | R-0/W1S | 0h | SD Module Interrupt Flag Clear Bits:<br>Writing a '1' will clear the respective flag bit in the SDINTFLG register. Writes of '0' are ignored.<br>Note: If user writes a '1' to clear a bit on the same cycle that the hardware is trying to set the bit to '1', then hardware has priority and the bit will not be cleared.<br>Flag-clear bit for Acknowledge flag for Filter 4<br>Reset type: SYSRSn |
| 14 | AF3 | R-0/W1S | 0h | Flag-clear bit for Acknowledge flag for Filter 3<br>Reset type: SYSRSn |
| 13 | AF2 | R-0/W1S | 0h | Flag-clear bit for Acknowledge flag for Filter 2<br>Reset type: SYSRSn |
| 12 | AF1 | R-0/W1S | 0h | Flag-clear bit for Acknowledge flag for Filter 1<br>Reset type: SYSRSn |
| 11 | MF4 | R-0/W1S | 0h | Flag-clear bit for Modulator Failure for Filter 4<br>Reset type: SYSRSn |
| 10 | MF3 | R-0/W1S | 0h | Flag-clear bit for Modulator Failure for Filter 3<br>Reset type: SYSRSn |
| 9 | MF2 | R-0/W1S | 0h | Flag-clear bit for Modulator Failure for Filter 2<br>Reset type: SYSRSn |
| 8 | MF1 | R-0/W1S | 0h | Flag-clear bit for Modulator Failure for Filter 1<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 14-16. SDIFLGCLR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 7 | IFL4 | R-0/W1S | 0h | Flag-clear bit for Low-Level Interrupt flag Filter 4<br>Reset type: SYSRSn |
| 6 | IFH4 | R-0/W1S | 0h | Flag-clear bit for High-level Interrupt flag Filter 4<br>Reset type: SYSRSn |
| 5 | IFL3 | R-0/W1S | 0h | Flag-clear bit for Low-Level Interrupt flag Filter 3<br>Reset type: SYSRSn |
| 4 | IFH3 | R-0/W1S | 0h | Flag-clear bit for High-level Interrupt flag Filter 3<br>Reset type: SYSRSn |
| 3 | IFL2 | R-0/W1S | 0h | Flag-clear bit for Low-Level Interrupt flag Filter 2<br>Reset type: SYSRSn |
| 2 | IFH2 | R-0/W1S | 0h | Flag-clear bit for High-level Interrupt flag Filter 2<br>Reset type: SYSRSn |
| 1 | IFL1 | R-0/W1S | 0h | Flag-clear bit for Low-Level Interrupt flag Filter 1<br>Reset type: SYSRSn |
| 0 | IFH1 | R-0/W1S | 0h | Flag-clear bit for High-level Interrupt flag Filter 1<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 14.11.2.3 SDCTL Register (Offset = 4h) [Reset = 0000h]

SDCTL is shown in Figure 14-13 and described in Table 14-17.

Return to the Summary Table.

SD Control Register

**Figure 14-13. SDCTL Register**

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

**Table 14-17. SDCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RESERVED | R-0 | 0h | Reserved |
| 14 | RESERVED | R-0 | 0h | Reserved |
| 13 | MIE | R/W | 0h | Master interrupt enable.<br>0: Interrupt pin and interrupt flags are blocked (interrupt pin INT always inactive).<br>1: Interrupt pin and interrupt flags are not blocked and can be set and reset (if individually<br>enabled).<br>Reset type: SYSRSn |
| 12-0 | RESERVED | R-0 | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 14.11.2.4 SDMFILEN Register (Offset = 6h) [Reset = 0000h]

SDMFILEN is shown in Figure 14-14 and described in Table 14-18.

Return to the Summary Table.

SD Master Filter Enable

**Figure 14-14. SDMFILEN Register**

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

**Table 14-18. SDMFILEN Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-13 | RESERVED | R-0 | 0h | Reserved |
| 12 | RESERVED | R-0 | 0h | Reserved |
| 11 | MFE | R/W | 0h | Master Filter Enable. Functionally AND'ed with bit FEN in the Data Filter Parameter Register<br>0: Data filter units of all filter modules are disabled.<br>1: Data filter units can be enabled if bit FEN is '1'.<br>Reset type: SYSRSn |
| 10 | RESERVED | R-0 | 0h | Reserved |
| 9 | RESERVED | R-0 | 0h | Reserved |
| 8-7 | RESERVED | R-0 | 0h | Reserved |
| 6-4 | RESERVED | R-0 | 0h | Reserved |
| 3-0 | RESERVED | R-0 | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 14.11.2.5 SDCTLPARM1 Register (Offset = 10h) [Reset = 0000h]

SDCTLPARM1 is shown in Figure 14-15 and described in Table 14-19.

Return to the Summary Table.

Control Parameter Register for Ch1

**Figure 14-15. SDCTLPARM1 Register**

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

**Table 14-19. SDCTLPARM1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-5 | RESERVED | R | 0h | Reserved |
| 4 | RESERVED | R-0 | 0h | Reserved |
| 3 | RESERVED | R-0 | 0h | Reserved |
| 2 | RESERVED | R-0 | 0h | Reserved |
| 1-0 | MOD | R/W | 0h | Delta-Sigma Modulator mode<br>00: The clock speed is equal to the data rate from the modulator<br>01: The clock rate is half of the data rate from the modulator<br>10: The data from the modulator is Manchester decoded<br>11: The clock rate is twice the data rate of the modulator<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 14.11.2.6 SDDFPARM1 Register (Offset = 11h) [Reset = 0000h]

SDDFPARM1 is shown in Figure 14-16 and described in Table 14-20.

Return to the Summary Table.

Data Filter Parameter Register for Ch1

**Figure 14-16. SDDFPARM1 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | SDSYNCEN | SST | | AE | FEN |
| R-0h | | | R/W-0h | R/W-0h | | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DOSR | | | | | | | |
| R/W-0h | | | | | | | |

**Table 14-20. SDDFPARM1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-13 | RESERVED | R | 0h | Reserved |
| 12 | SDSYNCEN | R/W | 0h | Data Filter Reset enable for External Reset typ from PWM Compare output.<br>0: Data filter cannot be reset by external PWM compare output<br>1: Data filter can be reset by external PWM compare output<br>Reset type: SYSRSn |
| 11-10 | SST | R/W | 0h | Data filter structure.<br>00: Data filter runs with a Sincfast structure<br>01: Data filter runs with a Sinc1 structure<br>10: Data filter runs with a Sinc2 structure<br>11: Data filter runs with a Sinc3 structure<br>Reset type: SYSRSn |
| 9 | AE | R/W | 0h | Acknowledge enable.<br>0: Acknowledge flag is disabled for the particular filter<br>1: Acknowledge flag is enabled for the particular filter<br>Reset type: SYSRSn |
| 8 | FEN | R/W | 0h | Filter enable.<br>0: The filter is disabled and no data is produced<br>1: The filter is enabled and data are produced in the Data filter<br>Reset type: SYSRSn |
| 7-0 | DOSR | R/W | 0h | Oversampling ratio. The actual rate is DOSR + 1.<br>These bits set the oversampling ratio of the filter.<br>0x0FF represents an oversampling ratio of 256.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 14.11.2.7 SDDPARM1 Register (Offset = 12h) [Reset = 0000h]

SDDPARM1 is shown in Figure 14-17 and described in Table 14-21.

Return to the Summary Table.

Integer Parameter Register for Ch1

**Figure 14-17. SDDPARM1 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| SH | | | | | DR | RESERVED | RESERVED |
| R/W-0h | | | | | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | RESERVED | | | | | | |
| R-0-0h | R/W-0h | | | | | | |

**Table 14-21. SDDPARM1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-11 | SH | R/W | 0h | Shift Control<br>These bits indicate by how many bits the 16-bit window is shifted up when 16-bit data representation is chosen.<br>Reset type: SYSRSn |
| 10 | DR | R/W | 0h | Data representation<br>0: Data stored in 16b 2's complement<br>1: Data stored in 32b 2's complement<br>Reset type: SYSRSn |
| 9 | RESERVED | R/W | 0h | Reserved |
| 8 | RESERVED | R/W | 0h | Reserved |
| 7 | RESERVED | R-0 | 0h | Reserved |
| 6-0 | RESERVED | R/W | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 14.11.2.8 SDCMPH1 Register (Offset = 13h) [Reset = 0000h]

SDCMPH1 is shown in Figure 14-18 and described in Table 14-22.

Return to the Summary Table.

High-level Threshold Register for Ch1

**Figure 14-18. SDCMPH1 Register**

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

**Table 14-22. SDCMPH1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RESERVED | R-0 | 0h | Reserved |
| 14-0 | HLT | R/W | 0h | Unsigned high-level threshold for the comparator filter output.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 14.11.2.9 SDCMPL1 Register (Offset = 14h) [Reset = 0000h]

SDCMPL1 is shown in Figure 14-19 and described in Table 14-23.

Return to the Summary Table.

Low-level Threshold Register for Ch1

**Figure 14-19. SDCMPL1 Register**

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

**Table 14-23. SDCMPL1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RESERVED | R-0 | 0h | Reserved |
| 14-0 | LLT | R/W | 0h | Unsigned low-level threshold for the comparator filter output.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 14.11.2.10 SDCPARM1 Register (Offset = 15h) [Reset = 0000h]

SDCPARM1 is shown in Figure 14-20 and described in Table 14-24.

Return to the Summary Table.

Comparator Parameter Register for Ch1

**Figure 14-20. SDCPARM1 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | MFIE | CS1_CS0 |
| R-0-0h | | | | | | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CS1_CS0 | IEL | IEH | COSR | | | | |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | | | | |

**Table 14-24. SDCPARM1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-10 | RESERVED | R-0 | 0h | Reserved |
| 9 | MFIE | R/W | 0h | 0: The modulator failure flag as well as the output INT is disabled for this particular flag<br>1: The modulator failure flag is enabled<br>Reset type: SYSRSn |
| 8-7 | CS1_CS0 | R/W | 0h | Comparator filter structure.<br>00: Comparator filter runs with a sincfast structure<br>01: Comparator filter runs with a Sinc1 structure<br>10: Comparator filter runs with a Sinc2 structure<br>11: Comparator filter runs with a Sinc3 structure<br>Reset type: SYSRSn |
| 6 | IEL | R/W | 0h | Low-level interrupt enable.<br>0: The low-level interrupt flag as well as the output INT is disabled for this particular flag<br>1: The low-level interrupt flag is enabled<br>Reset type: SYSRSn |
| 5 | IEH | R/W | 0h | High-level interrupt enable.<br>0: The high-level interrupt flag as well as the output INT is disabled for this particular flag<br>1: The high-level interrupt flag is enabled<br>Reset type: SYSRSn |
| 4-0 | COSR | R/W | 0h | Oversampling ratio. The actual rate is COSR + 1.<br>These bits set the oversampling ratio of the filter.<br>0x1F represents an oversampling ratio of 31.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 14.11.2.11 SDDATA1 Register (Offset = 16h) [Reset = 00000000h]

SDDATA1 is shown in Figure 14-21 and described in Table 14-25.

Return to the Summary Table.

Filter Data Register (16 or 32bit) for Ch1

**Figure 14-21. SDDATA1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA32HI | | | | | | | | | | | | | | | | DATA16 | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | R-0h | | | | | | | | | | | | | | | |

**Table 14-25. SDDATA1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | DATA32HI | R | 0h | Hi-order 16b in 32b mode, 16-bit Data in 16b mode<br>Reset type: SYSRSn |
| 15-0 | DATA16 | R | 0h | Lo-order 16b in 32b mode<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 14.11.2.12 SDCTLPARM2 Register (Offset = 20h) [Reset = 0000h]

SDCTLPARM2 is shown in Figure 14-22 and described in Table 14-26.

Return to the Summary Table.

Control Parameter Register for Ch2

**Figure 14-22. SDCTLPARM2 Register**

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

**Table 14-26. SDCTLPARM2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-5 | RESERVED | R | 0h | Reserved |
| 4 | RESERVED | R-0 | 0h | Reserved |
| 3 | RESERVED | R-0 | 0h | Reserved |
| 2 | RESERVED | R-0 | 0h | Reserved |
| 1-0 | MOD | R/W | 0h | Delta-Sigma Modulator mode<br>00: The clock speed is equal to the data rate from the modulator<br>01: The clock rate is half of the data rate from the modulator<br>10: The data from the modulator is Manchester decoded<br>11: The clock rate is twice the data rate of the modulator<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 14.11.2.13 SDDFPARM2 Register (Offset = 21h) [Reset = 0000h]

SDDFPARM2 is shown in Figure 14-23 and described in Table 14-27.

Return to the Summary Table.

Data Filter Parameter Register for Ch2

**Figure 14-23. SDDFPARM2 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | SDSYNCEN | SST | | AE | FEN |
| R-0h | | | R/W-0h | R/W-0h | | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DOSR | | | | | | | |
| R/W-0h | | | | | | | |

**Table 14-27. SDDFPARM2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-13 | RESERVED | R | 0h | Reserved |
| 12 | SDSYNCEN | R/W | 0h | Data Filter Reset enable for External Reset typ from PWM Compare output.<br>0: Data filter cannot be reset by external PWM compare output<br>1: Data filter can be reset by external PWM compare output<br>Reset type: SYSRSn |
| 11-10 | SST | R/W | 0h | Data filter structure.<br>00: Data filter runs with a Sincfast structure<br>01: Data filter runs with a Sinc1 structure<br>10: Data filter runs with a Sinc2 structure<br>11: Data filter runs with a Sinc3 structure<br>Reset type: SYSRSn |
| 9 | AE | R/W | 0h | Acknowledge enable.<br>0: Acknowledge flag is disabled for the particular filter<br>1: Acknowledge flag is enabled for the particular filter<br>Reset type: SYSRSn |
| 8 | FEN | R/W | 0h | Filter enable.<br>0: The filter is disabled and no data is produced<br>1: The filter is enabled and data are produced in the Data filter<br>Reset type: SYSRSn |
| 7-0 | DOSR | R/W | 0h | Oversampling ratio. The actual rate is DOSR + 1.<br>These bits set the oversampling ratio of the filter.<br>0x0FF represents an oversampling ratio of 256.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 14.11.2.14 SDDPARM2 Register (Offset = 22h) [Reset = 0000h]

SDDPARM2 is shown in Figure 14-24 and described in Table 14-28.

Return to the Summary Table.

Integer Parameter Register for Ch2

**Figure 14-24. SDDPARM2 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| SH | | | | | DR | RESERVED | RESERVED |
| R/W-0h | | | | | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | RESERVED | | | | | | |
| R-0-0h | R/W-0h | | | | | | |

**Table 14-28. SDDPARM2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-11 | SH | R/W | 0h | Shift Control<br>These bits indicate by how many bits the 16-bit window is shifted up when 16-bit data representation is chosen.<br>Reset type: SYSRSn |
| 10 | DR | R/W | 0h | Data representation<br>0: Data stored in 16b 2's complement<br>1: Data stored in 32b 2's complement<br>Reset type: SYSRSn |
| 9 | RESERVED | R/W | 0h | Reserved |
| 8 | RESERVED | R/W | 0h | Reserved |
| 7 | RESERVED | R-0 | 0h | Reserved |
| 6-0 | RESERVED | R/W | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 14.11.2.15 SDCMPH2 Register (Offset = 23h) [Reset = 0000h]

SDCMPH2 is shown in Figure 14-25 and described in Table 14-29.

Return to the Summary Table.

High-level Threshold Register for Ch2

**Figure 14-25. SDCMPH2 Register**

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

**Table 14-29. SDCMPH2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RESERVED | R-0 | 0h | Reserved |
| 14-0 | HLT | R/W | 0h | Unsigned high-level threshold for the comparator filter output.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 14.11.2.16 SDCMPL2 Register (Offset = 24h) [Reset = 0000h]

SDCMPL2 is shown in Figure 14-26 and described in Table 14-30.

Return to the Summary Table.

Low-level Threshold Register for Ch2

**Figure 14-26. SDCMPL2 Register**

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

**Table 14-30. SDCMPL2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RESERVED | R-0 | 0h | Reserved |
| 14-0 | LLT | R/W | 0h | Unsigned low-level threshold for the comparator filter output.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 14.11.2.17 SDCPARM2 Register (Offset = 25h) [Reset = 0000h]

SDCPARM2 is shown in Figure 14-27 and described in Table 14-31.

Return to the Summary Table.

Comparator Parameter Register for Ch2

**Figure 14-27. SDCPARM2 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | MFIE | CS1_CS0 |
| R-0-0h | | | | | | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CS1_CS0 | IEL | IEH | COSR | | | | |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | | | | |

**Table 14-31. SDCPARM2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-10 | RESERVED | R-0 | 0h | Reserved |
| 9 | MFIE | R/W | 0h | 0: The modulator failure flag as well as the output INT is disabled for this particular flag<br>1: The modulator failure flag is enabled<br>Reset type: SYSRSn |
| 8-7 | CS1_CS0 | R/W | 0h | Comparator filter structure.<br>00: Comparator filter runs with a sincfast structure<br>01: Comparator filter runs with a Sinc1 structure<br>10: Comparator filter runs with a Sinc2 structure<br>11: Comparator filter runs with a Sinc3 structure<br>Reset type: SYSRSn |
| 6 | IEL | R/W | 0h | Low-level interrupt enable.<br>0: The low-level interrupt flag as well as the output INT is disabled for this particular flag<br>1: The low-level interrupt flag is enabled<br>Reset type: SYSRSn |
| 5 | IEH | R/W | 0h | High-level interrupt enable.<br>0: The high-level interrupt flag as well as the output INT is disabled for this particular flag<br>1: The high-level interrupt flag is enabled<br>Reset type: SYSRSn |
| 4-0 | COSR | R/W | 0h | Oversampling ratio. The actual rate is COSR + 1.<br>These bits set the oversampling ratio of the filter.<br>0x1F represents an oversampling ratio of 31.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 14.11.2.18 SDDATA2 Register (Offset = 26h) [Reset = 00000000h]

SDDATA2 is shown in Figure 14-28 and described in Table 14-32.

Return to the Summary Table.

Filter Data Register (16 or 32bit) for Ch2

**Figure 14-28. SDDATA2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA32HI | | | | | | | | | | | | | | | | DATA16 | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | R-0h | | | | | | | | | | | | | | | |

**Table 14-32. SDDATA2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | DATA32HI | R | 0h | Hi-order 16b in 32b mode, 16-bit Data in 16b mode<br>Reset type: SYSRSn |
| 15-0 | DATA16 | R | 0h | Lo-order 16b in 32b mode<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 14.11.2.19 SDCTLPARM3 Register (Offset = 30h) [Reset = 0000h]

SDCTLPARM3 is shown in Figure 14-29 and described in Table 14-33.

Return to the Summary Table.

Control Parameter Register for Ch3

**Figure 14-29. SDCTLPARM3 Register**

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

**Table 14-33. SDCTLPARM3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-5 | RESERVED | R | 0h | Reserved |
| 4 | RESERVED | R-0 | 0h | Reserved |
| 3 | RESERVED | R-0 | 0h | Reserved |
| 2 | RESERVED | R-0 | 0h | Reserved |
| 1-0 | MOD | R/W | 0h | Delta-Sigma Modulator mode<br>00: The clock speed is equal to the data rate from the modulator<br>01: The clock rate is half of the data rate from the modulator<br>10: The data from the modulator is Manchester decoded<br>11: The clock rate is twice the data rate of the modulator<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 14.11.2.20 SDDFPARM3 Register (Offset = 31h) [Reset = 0000h]

SDDFPARM3 is shown in Figure 14-30 and described in Table 14-34.

Return to the Summary Table.

Data Filter Parameter Register for Ch3

**Figure 14-30. SDDFPARM3 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | SDSYNCEN | SST | | AE | FEN |
| R-0h | | | R/W-0h | R/W-0h | | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DOSR | | | | | | | |
| R/W-0h | | | | | | | |

**Table 14-34. SDDFPARM3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-13 | RESERVED | R | 0h | Reserved |
| 12 | SDSYNCEN | R/W | 0h | Data Filter Reset enable for External Reset typ from PWM Compare output.<br>0: Data filter cannot be reset by external PWM compare output<br>1: Data filter can be reset by external PWM compare output<br>Reset type: SYSRSn |
| 11-10 | SST | R/W | 0h | Data filter structure.<br>00: Data filter runs with a Sincfast structure<br>01: Data filter runs with a Sinc1 structure<br>10: Data filter runs with a Sinc2 structure<br>11: Data filter runs with a Sinc3 structure<br>Reset type: SYSRSn |
| 9 | AE | R/W | 0h | Acknowledge enable.<br>0: Acknowledge flag is disabled for the particular filter<br>1: Acknowledge flag is enabled for the particular filter<br>Reset type: SYSRSn |
| 8 | FEN | R/W | 0h | Filter enable.<br>0: The filter is disabled and no data is produced<br>1: The filter is enabled and data are produced in the data filter<br>Reset type: SYSRSn |
| 7-0 | DOSR | R/W | 0h | Oversampling ratio. The actual rate is DOSR + 1.<br>These bits set the oversampling ratio of the filter.<br>0x0FF represents an oversampling ratio of 256.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 14.11.2.21 SDDPARM3 Register (Offset = 32h) [Reset = 0000h]

SDDPARM3 is shown in Figure 14-31 and described in Table 14-35.

Return to the Summary Table.

Integer Parameter Register for Ch3

**Figure 14-31. SDDPARM3 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| SH | | | | | DR | RESERVED | RESERVED |
| R/W-0h | | | | | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | RESERVED | | | | | | |
| R-0-0h | R/W-0h | | | | | | |

**Table 14-35. SDDPARM3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-11 | SH | R/W | 0h | Shift Control<br>These bits indicate by how many bits the 16-bit window is shifted up when 16-bit data representation is chosen.<br>Reset type: SYSRSn |
| 10 | DR | R/W | 0h | Data representation<br>0: Data stored in 16b 2's complement<br>1: Data stored in 32b 2's complement<br>Reset type: SYSRSn |
| 9 | RESERVED | R/W | 0h | Reserved |
| 8 | RESERVED | R/W | 0h | Reserved |
| 7 | RESERVED | R-0 | 0h | Reserved |
| 6-0 | RESERVED | R/W | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 14.11.2.22 SDCMPH3 Register (Offset = 33h) [Reset = 0000h]

SDCMPH3 is shown in Figure 14-32 and described in Table 14-36.

Return to the Summary Table.

High-level Threshold Register for Ch3

**Figure 14-32. SDCMPH3 Register**

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

**Table 14-36. SDCMPH3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RESERVED | R-0 | 0h | Reserved |
| 14-0 | HLT | R/W | 0h | Unsigned high-level threshold for the comparator filter output.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 14.11.2.23 SDCMPL3 Register (Offset = 34h) [Reset = 0000h]

SDCMPL3 is shown in Figure 14-33 and described in Table 14-37.

Return to the Summary Table.

Low-level Threshold Register for Ch3

**Figure 14-33. SDCMPL3 Register**

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

**Table 14-37. SDCMPL3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RESERVED | R-0 | 0h | Reserved |
| 14-0 | LLT | R/W | 0h | Unsigned low-level threshold for the comparator filter output.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 14.11.2.24 SDCPARM3 Register (Offset = 35h) [Reset = 0000h]

SDCPARM3 is shown in Figure 14-34 and described in Table 14-38.

Return to the Summary Table.

Comparator Parameter Register for Ch3

**Figure 14-34. SDCPARM3 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | MFIE | CS1_CS0 |
| R-0-0h | | | | | | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CS1_CS0 | IEL | IEH | COSR | | | | |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | | | | |

**Table 14-38. SDCPARM3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-10 | RESERVED | R-0 | 0h | Reserved |
| 9 | MFIE | R/W | 0h | 0: The modulator failure flag as well as the output INT is disabled for this particular flag<br>1: The modulator failure flag is enabled<br>Reset type: SYSRSn |
| 8-7 | CS1_CS0 | R/W | 0h | Comparator filter structure.<br>00: Comparator filter runs with a sincfast structure<br>01: Comparator filter runs with a Sinc1 structure<br>10: Comparator filter runs with a Sinc2 structure<br>11: Comparator filter runs with a Sinc3 structure<br>Reset type: SYSRSn |
| 6 | IEL | R/W | 0h | Low-level interrupt enable.<br>0: The low-level interrupt flag as well as the output INT is disabled for this particular flag<br>1: The low-level interrupt flag is enabled<br>Reset type: SYSRSn |
| 5 | IEH | R/W | 0h | High-level interrupt enable.<br>0: The high-level interrupt flag as well as the output INT is disabled for this particular flag<br>1: The high-level interrupt flag is enabled<br>Reset type: SYSRSn |
| 4-0 | COSR | R/W | 0h | Oversampling ratio. The actual rate is COSR + 1.<br>These bits set the oversampling ratio of the filter.<br>0x1F represents an oversampling ratio of 31.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 14.11.2.25 SDDATA3 Register (Offset = 36h) [Reset = 00000000h]

SDDATA3 is shown in Figure 14-35 and described in Table 14-39.

Return to the Summary Table.

Filter Data Register (16 or 32bit) for Ch3

**Figure 14-35. SDDATA3 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA32HI | | | | | | | | | | | | | | | | DATA16 | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | R-0h | | | | | | | | | | | | | | | |

**Table 14-39. SDDATA3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | DATA32HI | R | 0h | Hi-order 16b in 32b mode, 16-bit Data in 16b mode<br>Reset type: SYSRSn |
| 15-0 | DATA16 | R | 0h | Lo-order 16b in 32b mode<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 14.11.2.26 SDCTLPARM4 Register (Offset = 40h) [Reset = 0000h]

SDCTLPARM4 is shown in Figure 14-36 and described in Table 14-40.

Return to the Summary Table.

Control Parameter Register for Ch4

**Figure 14-36. SDCTLPARM4 Register**

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

**Table 14-40. SDCTLPARM4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-5 | RESERVED | R | 0h | Reserved |
| 4 | RESERVED | R-0 | 0h | Reserved |
| 3 | RESERVED | R-0 | 0h | Reserved |
| 2 | RESERVED | R-0 | 0h | Reserved |
| 1-0 | MOD | R/W | 0h | Delta-Sigma Modulator mode<br>00: The clock speed is equal to the data rate from the modulator<br>01: The clock rate is half of the data rate from the modulator<br>10: The data from the modulator is Manchester decoded<br>11: The clock rate is twice the data rate of the modulator<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 14.11.2.27 SDDFPARM4 Register (Offset = 41h) [Reset = 0000h]

SDDFPARM4 is shown in Figure 14-37 and described in Table 14-41.

Return to the Summary Table.

Data Filter Parameter Register for Ch4

**Figure 14-37. SDDFPARM4 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | SDSYNCEN | SST | | AE | FEN |
| R-0h | | | R/W-0h | R/W-0h | | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DOSR | | | | | | | |
| R/W-0h | | | | | | | |

**Table 14-41. SDDFPARM4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-13 | RESERVED | R | 0h | Reserved |
| 12 | SDSYNCEN | R/W | 0h | Data Filter Reset enable for External Reset typ from PWM Compare output.<br>0: Data filter cannot be reset by external PWM compare output<br>1: Data filter can be reset by external PWM compare output<br>Reset type: SYSRSn |
| 11-10 | SST | R/W | 0h | Data filter structure.<br>00: Data filter runs with a Sincfast structure<br>01: Data filter runs with a Sinc1 structure<br>10: Data filter runs with a Sinc2 structure<br>11: Data filter runs with a Sinc3 structure<br>Reset type: SYSRSn |
| 9 | AE | R/W | 0h | Acknowledge enable.<br>0: Acknowledge flag is disabled for the particular filter<br>1: Acknowledge flag is enabled for the particular filter<br>Reset type: SYSRSn |
| 8 | FEN | R/W | 0h | Filter enable.<br>0: The filter is disabled and no data is produced<br>1: The filter is enabled and data are produced in the sinc filter<br>Reset type: SYSRSn |
| 7-0 | DOSR | R/W | 0h | Oversampling ratio. The actual rate is DOSR + 1.<br>These bits set the oversampling ratio of the filter.<br>0x0FF represents an oversampling ratio of 256.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 14.11.2.28 SDDPARM4 Register (Offset = 42h) [Reset = 0000h]

SDDPARM4 is shown in Figure 14-38 and described in Table 14-42.

Return to the Summary Table.

Integer Parameter Register for Ch4

**Figure 14-38. SDDPARM4 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| SH | | | | | DR | RESERVED | RESERVED |
| R/W-0h | | | | | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | RESERVED | | | | | | |
| R-0-0h | R/W-0h | | | | | | |

**Table 14-42. SDDPARM4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-11 | SH | R/W | 0h | Shift Control<br>These bits indicate by how many bits the 16-bit window is shifted up when 16-bit data representation is chosen.<br>Reset type: SYSRSn |
| 10 | DR | R/W | 0h | Data representation<br>0: Data stored in 16b 2's complement<br>1: Data stored in 32b 2's complement<br>Reset type: SYSRSn |
| 9 | RESERVED | R/W | 0h | Reserved |
| 8 | RESERVED | R/W | 0h | Reserved |
| 7 | RESERVED | R-0 | 0h | Reserved |
| 6-0 | RESERVED | R/W | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 14.11.2.29 SDCMPH4 Register (Offset = 43h) [Reset = 0000h]

SDCMPH4 is shown in Figure 14-39 and described in Table 14-43.

Return to the Summary Table.

High-level Threshold Register for Ch4

**Figure 14-39. SDCMPH4 Register**

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

**Table 14-43. SDCMPH4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RESERVED | R-0 | 0h | Reserved |
| 14-0 | HLT | R/W | 0h | Unsigned high-level threshold for the comparator filter output.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 14.11.2.30 SDCMPL4 Register (Offset = 44h) [Reset = 0000h]

SDCMPL4 is shown in Figure 14-40 and described in Table 14-44.

Return to the Summary Table.

Low-level Threshold Register for Ch4

**Figure 14-40. SDCMPL4 Register**

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

**Table 14-44. SDCMPL4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RESERVED | R-0 | 0h | Reserved |
| 14-0 | LLT | R/W | 0h | Unsigned low-level threshold for the comparator filter output.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 14.11.2.31 SDCPARM4 Register (Offset = 45h) [Reset = 0000h]

SDCPARM4 is shown in Figure 14-41 and described in Table 14-45.

Return to the Summary Table.

Comparator Parameter Register for Ch4

**Figure 14-41. SDCPARM4 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | MFIE | CS1_CS0 |
| R-0-0h | | | | | | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CS1_CS0 | IEL | IEH | COSR | | | | |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | | | | |

**Table 14-45. SDCPARM4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-10 | RESERVED | R-0 | 0h | Reserved |
| 9 | MFIE | R/W | 0h | 0: The modulator failure flag as well as the output INT is disabled for this particular flag<br>1: The modulator failure flag is enabled<br>Reset type: SYSRSn |
| 8-7 | CS1_CS0 | R/W | 0h | Comparator filter structure.<br>00: Comparator filter runs with a sincfast structure<br>01: Comparator filter runs with a Sinc1 structure<br>10: Comparator filter runs with a Sinc2 structure<br>11: Comparator filter runs with a Sinc3 structure<br>Reset type: SYSRSn |
| 6 | IEL | R/W | 0h | Low-level interrupt enable.<br>0: The low-level interrupt flag as well as the output INT is disabled for this particular flag<br>1: The low-level interrupt flag is enabled<br>Reset type: SYSRSn |
| 5 | IEH | R/W | 0h | High-level interrupt enable.<br>0: The high-level interrupt flag as well as the output INT is disabled for this particular flag<br>1: The high-level interrupt flag is enabled<br>Reset type: SYSRSn |
| 4-0 | COSR | R/W | 0h | Oversampling ratio. The actual rate is COSR + 1.<br>These bits set the oversampling ratio of the filter.<br>0x1F represents an oversampling ratio of 31.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 14.11.2.32 SDDATA4 Register (Offset = 46h) [Reset = 00000000h]

SDDATA4 is shown in Figure 14-42 and described in Table 14-46.

Return to the Summary Table.

Filter Data Register (16 or 32bit) for Ch4

**Figure 14-42. SDDATA4 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA32HI | | | | | | | | | | | | | | | | DATA16 | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | R-0h | | | | | | | | | | | | | | | |

**Table 14-46. SDDATA4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | DATA32HI | R | 0h | Hi-order 16b in 32b mode, 16-bit Data in 16b mode<br>Reset type: SYSRSn |
| 15-0 | DATA16 | R | 0h | Lo-order 16b in 32b mode<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 14.11.3 SDFM Registers to Driverlib Functions

**Table 14-47. SDFM Registers to Driverlib Functions**

| File | Driverlib Function |
|---|---|
| **SDIFLG** | |
| sdfm.h | SDFM_getThresholdStatus |
| sdfm.h | SDFM_getModulatorStatus |
| sdfm.h | SDFM_getNewFilterDataStatus |
| sdfm.h | SDFM_getIsrStatus |
| sdfm.h | SDFM_clearInterruptFlag |
| **SDIFLGCLR** | |
| sdfm.h | SDFM_clearInterruptFlag |
| **SDCTL** | |
| sdfm.h | SDFM_setupModulatorClock |
| sdfm.h | SDFM_enableMainInterrupt |
| sdfm.h | SDFM_disableMainInterrupt |
| **SDMFILEN** | |
| sdfm.h | SDFM_enableMainFilter |
| sdfm.h | SDFM_disableMainFilter |
| **SDCTLPARM1** | |
| sdfm.h | SDFM_setupModulatorClock |
| **SDDFPARM1** | |
| sdfm.h | SDFM_enableExternalReset |
| sdfm.h | SDFM_disableExternalReset |
| sdfm.h | SDFM_enableFilter |
| sdfm.h | SDFM_disableFilter |
| sdfm.h | SDFM_setFilterType |
| sdfm.h | SDFM_setFilterOverSamplingRatio |
| sdfm.h | SDFM_enableInterrupt |
| sdfm.h | SDFM_disableInterrupt |
| **SDDPARM1** | |
| sdfm.h | SDFM_setOutputDataFormat |
| sdfm.h | SDFM_setDataShiftValue |
| **SDCMPH1** | |
| sdfm.h | SDFM_setCompFilterHighThreshold |
| **SDCMPL1** | |
| sdfm.h | SDFM_setCompFilterLowThreshold |
| **SDCPARM1** | |
| sdfm.h | SDFM_enableInterrupt |
| sdfm.h | SDFM_disableInterrupt |
| sdfm.h | SDFM_setComparatorFilterType |
| sdfm.h | SDFM_setCompFilterOverSamplingRatio |
| **SDDATA1** | |
| sdfm.h | SDFM_getFilterData |
| **SDCTLPARM2** | |
| - | See SDCTLPARM1 |
| **SDDFPARM2** | |



Copyright © 2024 Texas Instruments Incorporated

**Table 14-47. SDFM Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| - | See SDDFPARM1 |
| **SDDPARM2** | |
| - | See SDDPARM1 |
| **SDCMPH2** | |
| - | See SDCMPH1 |
| **SDCMPL2** | |
| - | See SDCMPL1 |
| **SDCPARM2** | |
| - | See SDCPARM1 |
| **SDDATA2** | |
| - | See SDDATA1 |
| **SDCTLPARM3** | |
| - | See SDCTLPARM1 |
| **SDDFPARM3** | |
| - | See SDDFPARM1 |
| **SDDPARM3** | |
| - | See SDDPARM1 |
| **SDCMPH3** | |
| - | See SDCMPH1 |
| **SDCMPL3** | |
| - | See SDCMPL1 |
| **SDCPARM3** | |
| - | See SDCPARM1 |
| **SDDATA3** | |
| - | See SDDATA1 |
| **SDCTLPARM4** | |
| - | See SDCTLPARM1 |
| **SDDFPARM4** | |
| - | See SDDFPARM1 |
| **SDDPARM4** | |
| - | See SDDPARM1 |
| **SDCMPH4** | |
| - | See SDCMPH1 |
| **SDCMPL4** | |
| - | See SDCMPL1 |
| **SDCPARM4** | |
| - | See SDCPARM1 |
| **SDDATA4** | |
| - | See SDDATA1 |



Copyright © 2024 Texas Instruments Incorporated

Chapter 15

# 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.

Further information can be found in the *Flexible PWMs Enable Multi-Axis Drives, Multi-Level Inverters Technical Brief*.


**15.1 Introduction** ..... 1902
**15.2 Configuring Device Pins** ..... 1909
**15.3 ePWM Modules Overview** ..... 1909
**15.4 Time-Base (TB) Submodule** ..... 1911
**15.5 Counter-Compare (CC) Submodule** ..... 1924
**15.6 Action-Qualifier (AQ) Submodule** ..... 1930
**15.7 Dead-Band Generator (DB) Submodule** ..... 1943
**15.8 PWM Chopper (PC) Submodule** ..... 1950
**15.9 Trip-Zone (TZ) Submodule** ..... 1954
**15.10 Event-Trigger (ET) Submodule** ..... 1960
**15.11 Digital Compare (DC) Submodule** ..... 1965
**15.12 ePWM Crossbar (X-BAR)** ..... 1975
**15.13 Applications to Power Topologies** ..... 1976
**15.14 High-Resolution Pulse Width Modulator (HRPWM)** ..... 1994
**15.15 ePWM Registers** ..... 2020





Copyright © 2024 Texas Instruments Incorporated

## 15.1 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 15.14.
- **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.



Copyright © 2024 Texas Instruments Incorporated

- **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 15.14.
- **Digital Compare Submodule:** The digital compare submodule enhances the event triggering and trip zone submodules by providing filtering, blanking and improved trip functionality to digital compare signals. Such features are essential for peak current mode control and for support of analog comparators.

**Note**

The name of the sync signal that goes to the CMPSS and GPDAC has been updated from PWMSYNC to EPWMSYNCPER (SYNCPER/PWMSYNCPER/EPWMxSYNCPER) to avoid confusion with the other EPWM sync signals EPWMSYNCI and EPWMSYNCO. For a description of these signals, see Table 15-2.

### *15.1.1 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*



Copyright © 2024 Texas Instruments Incorporated

  - 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*

### *15.1.2 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 15-1. 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 15.14. 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:

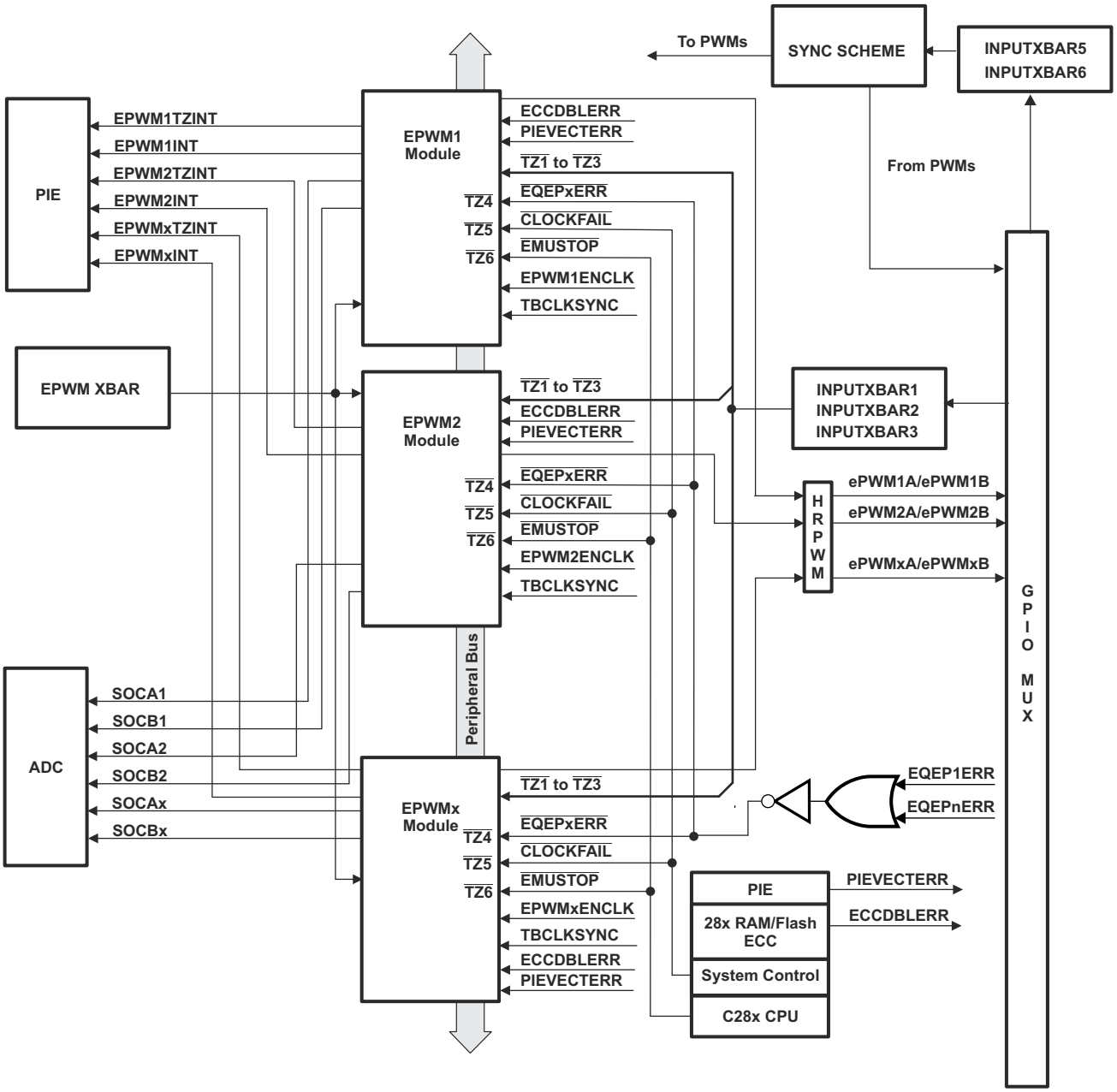
- Dedicated 16-bit time-base counter with period and frequency control
- Two PWM outputs (EPWMxA and EPWMxB) that can be used in the following configurations:
  - Two independent PWM outputs with single-edge operation
  - Two independent PWM outputs with dual-edge symmetric operation
  - One independent 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 15-1. 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 15-1. See Section 15.4.3.3 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 15-2.



Copyright © 2024 Texas Instruments Incorporated




A. This signal exists only on devices with an eQEP submodule.

**Figure 15-1. Multiple ePWM Modules**



Copyright © 2024 Texas Instruments Incorporated

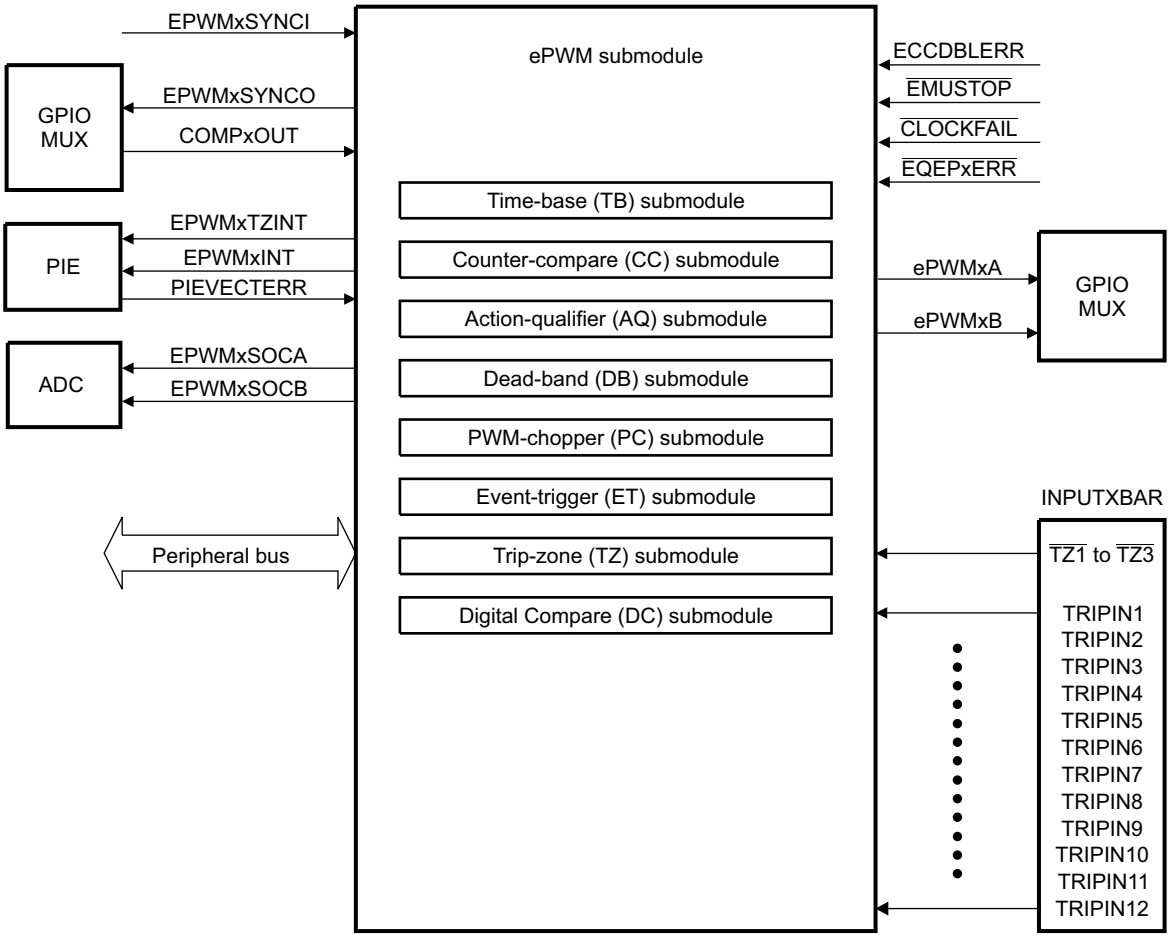



**Figure 15-2. Submodules and Signal Connections for an ePWM Module**



Copyright © 2024 Texas Instruments Incorporated

Figure 15-3 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 15-50. $\overline{TZ4}$ 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 $\overline{EMUSTOP}$ output from the CPU. This allows configuring a trip action when the clock fails or the CPU halts.

- **Time-base synchronization input (EPWMxSYNCI), output (EPWMxSYNCO), and peripheral (EPWMxSYNCPER) signals**

  The synchronization signals daisy chain the ePWM modules together. Each module can be configured to either use or ignore the synchronization input. The clock synchronization input and output signal are brought out to pins only for ePWM1 (ePWM module #1). The ePWM modules are separated into groups of three for syncing purposes. An external sync signal (EXTSYNCIN1 or EXTSYNCIN2) can be used to issue a sync signal to the first ePWM module in each chain. These same modules can also send the EPWMxSYNCOUT signal to a GPIO. For more information, see Section 15.4.3.3.

  Each ePWM module also generates another PWMSYNC signal called EPWMxSYNCPER. EPWMxSYNCPER goes to the GPDAC and 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 GPDAC and 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)**

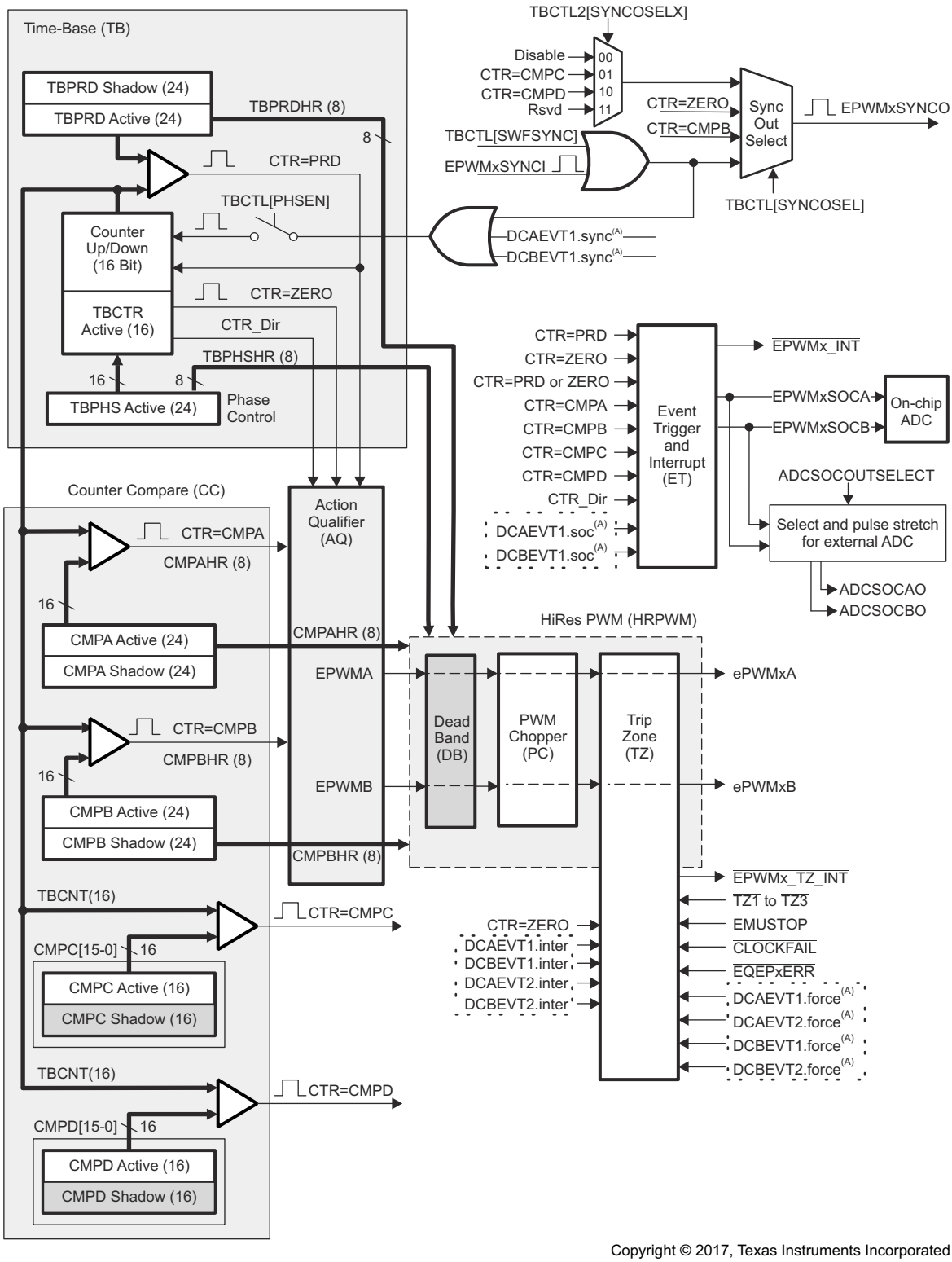
  Output signals from the comparator module can be fed through the Input X-BAR and EPWM X-BAR to one or all of the 12 trip inputs [TRIPIN1 - TRIPIN12] and in conjunction with the trip zone signals can generate digital compare events.

- **Peripheral bus**

  The peripheral bus is 32-bits wide and allows both 16-bit and 32-bit writes to the ePWM register file.



Copyright © 2024 Texas Instruments Incorporated




Copyright © 2017, Texas Instruments Incorporated

A. These events are generated by the ePWM Digital Compare (DC) submodule based on the levels of the TRIPIN inputs.

**Figure 15-3. ePWM Modules and Critical Internal Signal Interconnects**



Copyright © 2024 Texas Instruments Incorporated

## 15.2 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.

## 15.3 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 15-1 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 15.5 for relevant details.

**Table 15-1. Submodule Configuration Parameters**

| Submodule | Configuration Parameter or Option |
|---|---|
| Time Base (TB) | • Scale the time-base clock (TBCLK) relative to the ePWM clock (EPWMCLK).<br>• Configure the PWM time-base counter (TBCTR) frequency or period.<br>• Set the mode for the time-base counter:<br>– count-up mode: used for asymmetric PWM<br>– count-down mode: used for asymmetric PWM<br>– count-up-and-down mode: used for symmetric PWM<br>• Configure the time-base phase relative to another ePWM module.<br>• Synchronize the time-base counter between modules through hardware or software.<br>• Configure the direction (up or down) of the time-base counter after a synchronization event.<br>• Simultaneous writes to the TBPRD registers on all PWM's corresponding to the configuration on EPWMXLINK.<br>• Configure how the time-base counter behaves when the device is halted by an emulator.<br>• Specify the source for the synchronization output of the ePWM module<br>– Synchronization input signal<br>– Time-base counter equal to zero<br>– Time-base counter equal to counter-compare B (CMPB)<br>– No output synchronization signal generated.<br>• Configure one shot and global load of registers in this module. |
| Counter Compare (CC) | • Specify the PWM duty cycle for output EPWMxA and output EPWMxB<br>• Specify the time at which switching events occur on the EPWMxA or EPWMxB output<br>• Specify the programmable delay for interrupt and SOC generation with additional comparators<br>• Simultaneous writes to the CMPA, CMPB, CMPC, CMPD registers on all PWM's corresponding to the configuration on EPWMXLINK.<br>• Configure one shot and global load of registers in this module. |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-1. Submodule Configuration Parameters (continued)**

| Submodule | Configuration Parameter or Option |
|---|---|
| Action Qualifier (AQ) | • Specify the type of action taken when a time-base counter-compare, trip-zone submodule, or comparator event occurs:<br>– No action taken<br>– Output EPWMxA and EPWMxB switched high<br>– Output EPWMxA and EPWMxB switched low<br>– Output EPWMxA and EPWMxB toggled<br>• Force the PWM output state through software control<br>• Configure and control the PWM dead band through software<br>• Configure one shot and global load of registers in this module. |
| Dead-Band Generator (DB) | • Control of traditional complementary dead-band relationship between upper and lower switches<br>• Specify the output rising-edge-delay value<br>• Specify the output falling-edge delay value<br>• Bypass the dead-band module entirely. In this case the PWM waveform is passed through without modification.<br>• Option to enable half-cycle clocking for double resolution.<br>• Allow ePWMxB phase shifting with respect to the ePWMxA output.<br>• Configure one shot and global load of registers in this module. |
| PWM Chopper (PC) | • Create a chopping (carrier) frequency.<br>• Pulse width of the first pulse in the chopped pulse train.<br>• Duty cycle of the second and subsequent pulses.<br>• Bypass the PWM chopper module entirely. In this case the PWM waveform is passed through without modification. |
| Trip Zone (TZ) | • Configure the ePWM module to react to one, all, or none of the trip-zone signals or digital compare events.<br>• Specify the trip action taken when a fault occurs:<br>– Force EPWMxA and EPWMxB high<br>– Force EPWMxA and EPWMxB low<br>– Force EPWMxA and EPWMxB to a high-impedance state<br>– Configure EPWMxA and EPWMxB to ignore any trip condition.<br>• Configure how often the ePWM reacts to each trip-zone signal:<br>– One-shot<br>– Cycle-by-cycle<br>• Enable the trip-zone to initiate an interrupt.<br>• Bypass the trip-zone module entirely.<br>• Programmable option for cycle-by-cycle trip clear<br>• If desired, independently configure trip actions taken when time-base counter is counting down. |
| Event Trigger (ET) | • Enable the ePWM events that trigger an interrupt.<br>• Enable ePWM events that trigger an ADC start-of-conversion event.<br>• Specify the rate at which events cause triggers (every occurrence or every 2nd or up to 15th occurrence)<br>• Poll, set, or clear event flags |
| Digital Compare (DC) | • Enables comparator (COMP) module outputs and trip zone signals which are configured using the Input X-BAR to create events and filtered events<br>• 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. |



Copyright © 2024 Texas Instruments Incorporated

## 15.4 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 15-4 illustrates the time-base submodule within the ePWM.

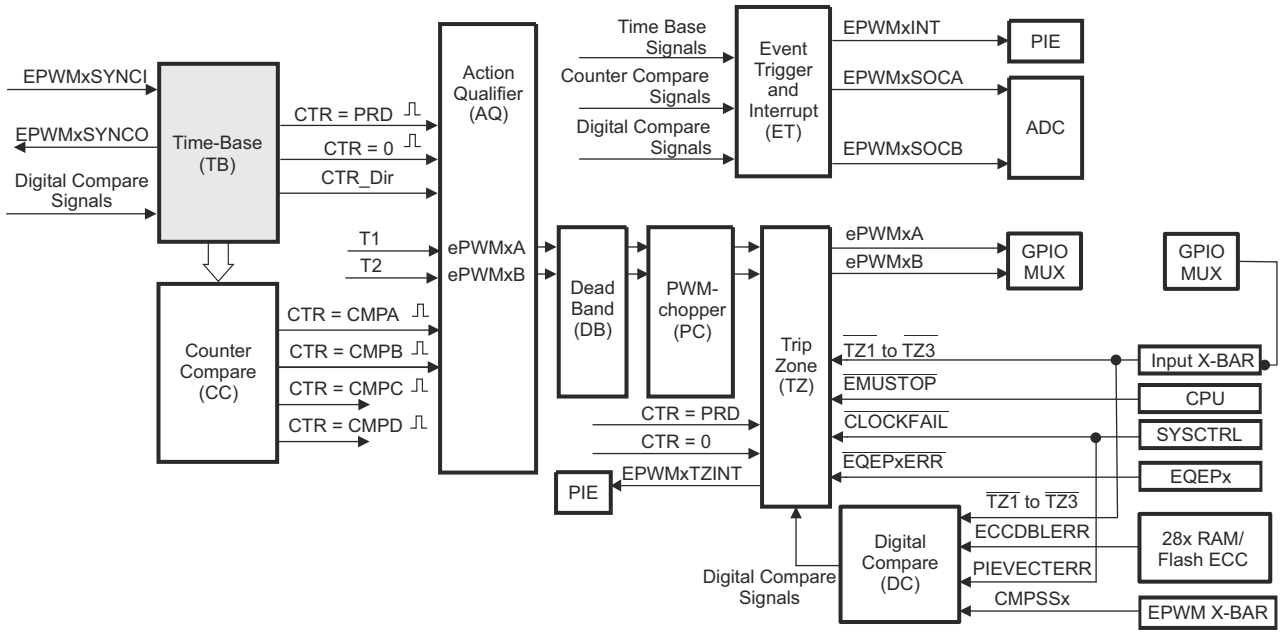



**Figure 15-4. Time-Base Submodule**

### *15.4.1 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**

The Type 4 ePWM clocking varies from previous ePWM types. Prior to the Type 4 ePWM, the time-base submodule is clocked directly by the system clock (SYSCLKOUT). On this version of the ePWM, there is a divider (EPWMCLKDIV) of the system clock that defaults to EPWMCLK = SYSCLKOUT/2.

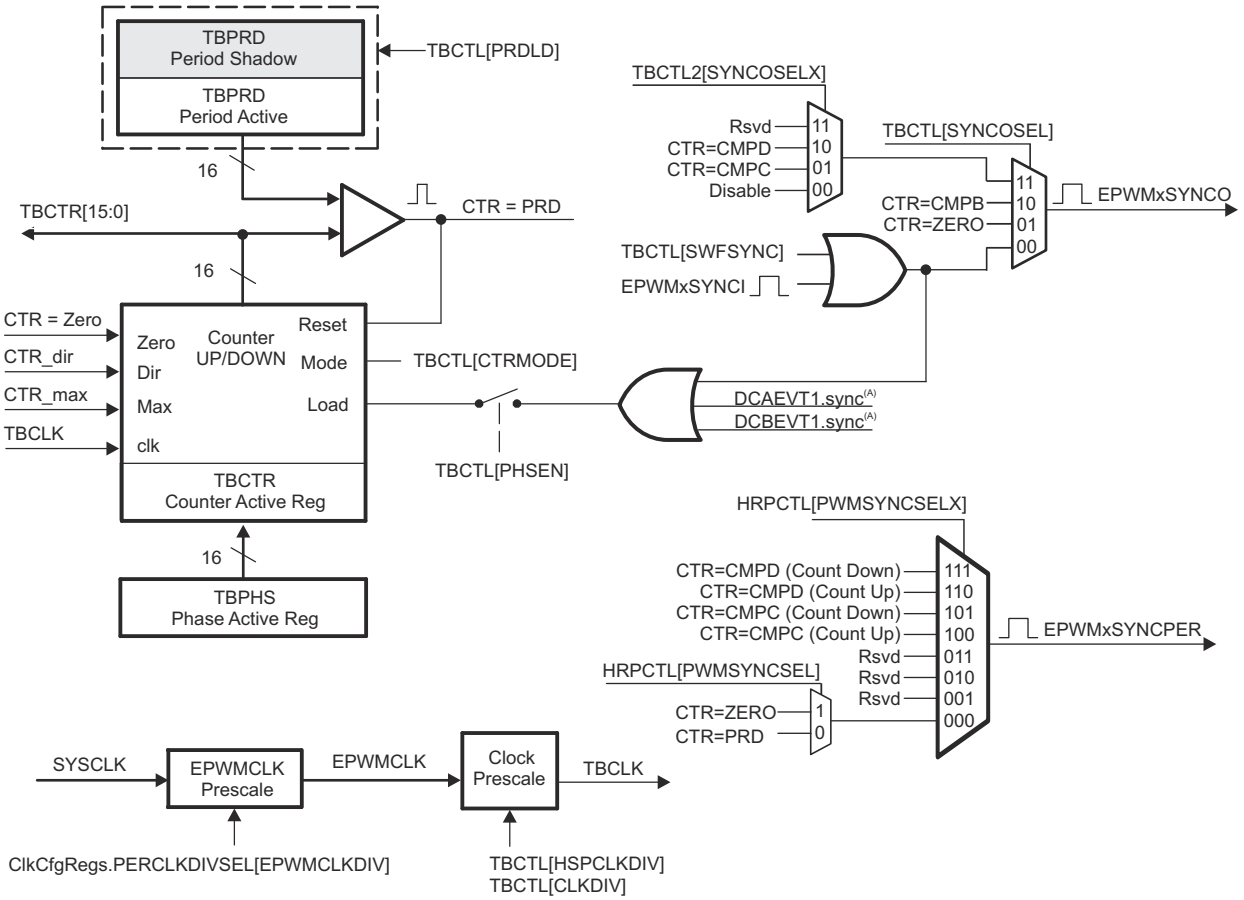
If required by the application code to update the TBCTR value through software while the TBCTR is counting, note that the time-base module needs at least 1 TBCLK cycle for the time-base related events to be realized. Hence, the TBCTR can be written with TBCTR = PRD-1 instead of TBCTR = PRD (in case the 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.



Copyright © 2024 Texas Instruments Incorporated

### 15.4.2 Controlling and Monitoring the Time-Base Submodule

The block diagram in Figure 15-5 shows the critical signals and registers of the time-base submodule. Table 15-2 provides descriptions of the key signals associated with the time-base submodule.




A. These signals are generated by the digital compare (DC) submodule.

**Figure 15-5. Time-Base Submodule Signals and Registers**



Copyright © 2024 Texas Instruments Incorporated

**Table 15-2. Key Time-Base Signals**

| Signal | Description |
|---|---|
| EPWMxSYNCI | Time-base synchronization input.<br>Input pulse used to synchronize the time-base counter with the counter of ePWM module earlier in the synchronization chain. An ePWM peripheral can be configured to use or ignore this signal. For the first ePWM module in each synchronization chain, this signal can come from a device pin using INPUT5 or INPUT6 of the Input X-BAR or from a previous ePWM module. For subsequent ePWM modules in each chain, this signal is passed from another ePWM peripheral. For example, EPWM2SYNCI is generated by the ePWM1 peripheral, EPWM3SYNCI is generated by ePWM2 and so forth. For information on the synchronization order of a particular device, see Section 15.4.3.3. |
| EPWMxSYNCO | Time-base synchronization output.<br>This output pulse is used to synchronize the counter of an ePWM module later in the synchronization chain. The ePWM module generates this signal from one of three event sources:<br>1. EPWMxSYNCI (Synchronization input pulse)<br>2. CTR = Zero: The time-base counter equal to zero (TBCTR = 0x00).<br>3. CTR = CMPB: The time-base counter equal to the counter-compare B (TBCTR = CMPB) register. |
| EPWMxSYNCPER | Time-base peripheral synchronization output.<br>This output signal is used to synchronize the GPDAC and 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.<br>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<br>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).<br>This event is generated by the counter-compare submodule and used by the synchronization out logic |
| CTR_dir | Time-base counter direction.<br>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)<br>Generated event when the TBCTR value reaches the maximum value. This signal is only used only as a status bit |
| TBCLK | Time-base clock.<br>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. |



Copyright © 2024 Texas Instruments Incorporated

### 15.4.3 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 15-6 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.

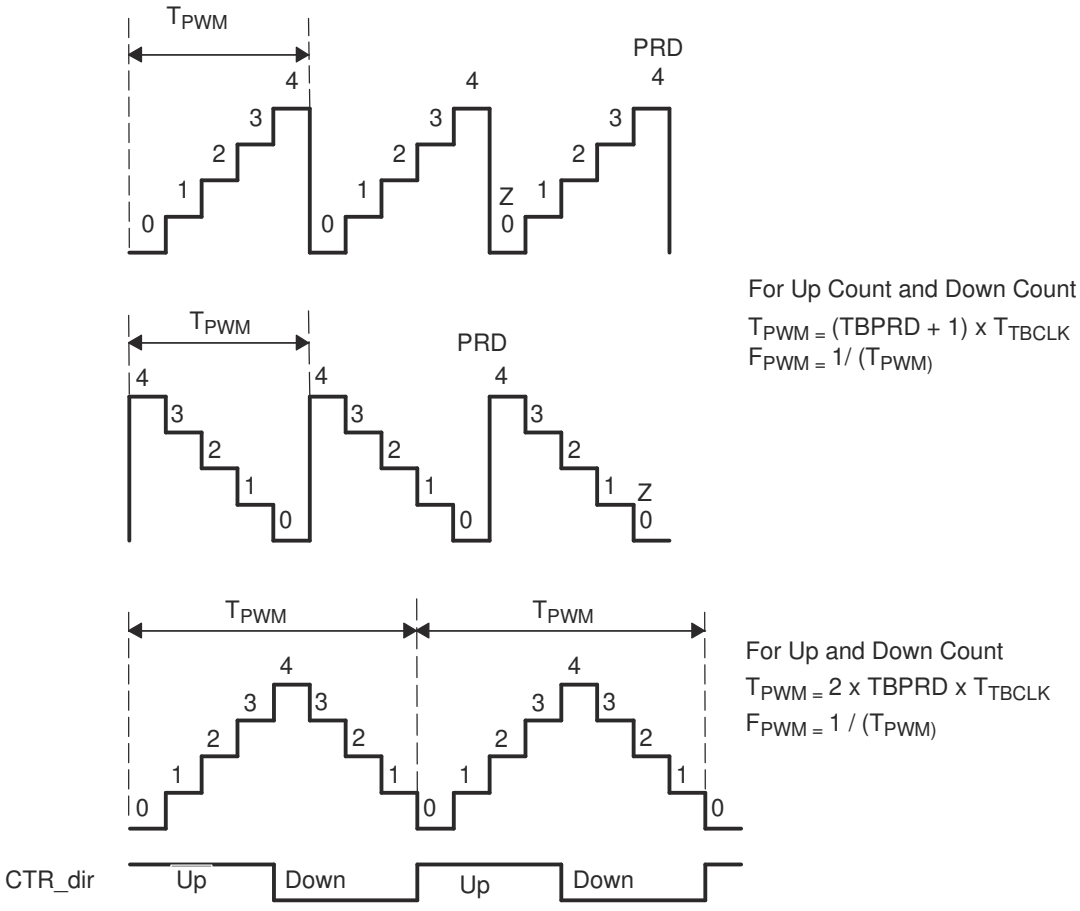



**Figure 15-6. Time-Base Frequency and Period**



Copyright © 2024 Texas Instruments Incorporated

#### 15.4.3.1 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[PRDLD] 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[PRDLD] = 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[PRDLD] = 0. By default the TBPRD shadow register is enabled. The sources for the SYNC input is explained in Section 15.4.3.3.

  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 15.4.7.
- **Time-Base Period Immediate Load Mode:** If immediate load mode is selected (TBCTL[PRDLD] = 1), then a read from or a write to the TBPRD memory address goes directly to the active register.

#### 15.4.3.2 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

In a multicore environment, GTBCLKSYNC can be used to override the core-specific TBCLKSYNC. When GTBCLKSYNC is set, TBCLKSYNC is ignored in all cores and therefore clearing or setting the TBCLKSYNC has no affect. If this feature is not required, GTBCLKSYNC must be cleared. In a multicore environment where different ePWM modules are assigned to different cores, the GTBCLKSYNC bit can be used to enable and disable the Time-Base clock of all ePWM modules simultaneously.



Copyright © 2024 Texas Instruments Incorporated

#### 15.4.3.3 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 (SYNCI), a synchronization output (SYNCO) and a peripheral synchronization output (SYNCPER). In Figure 15-7, EXTSYNCIN1 is sourced from INPUTXBAR5 and EXTSYNCIN2 is sourced from INPUTXBAR6, which can be configured to select any GPIO as the synchronization input. When configuring the sync chain propagation path using the SYNCSEL registers, make sure that the longest path does not exceed four ePWM/eCAP modules.

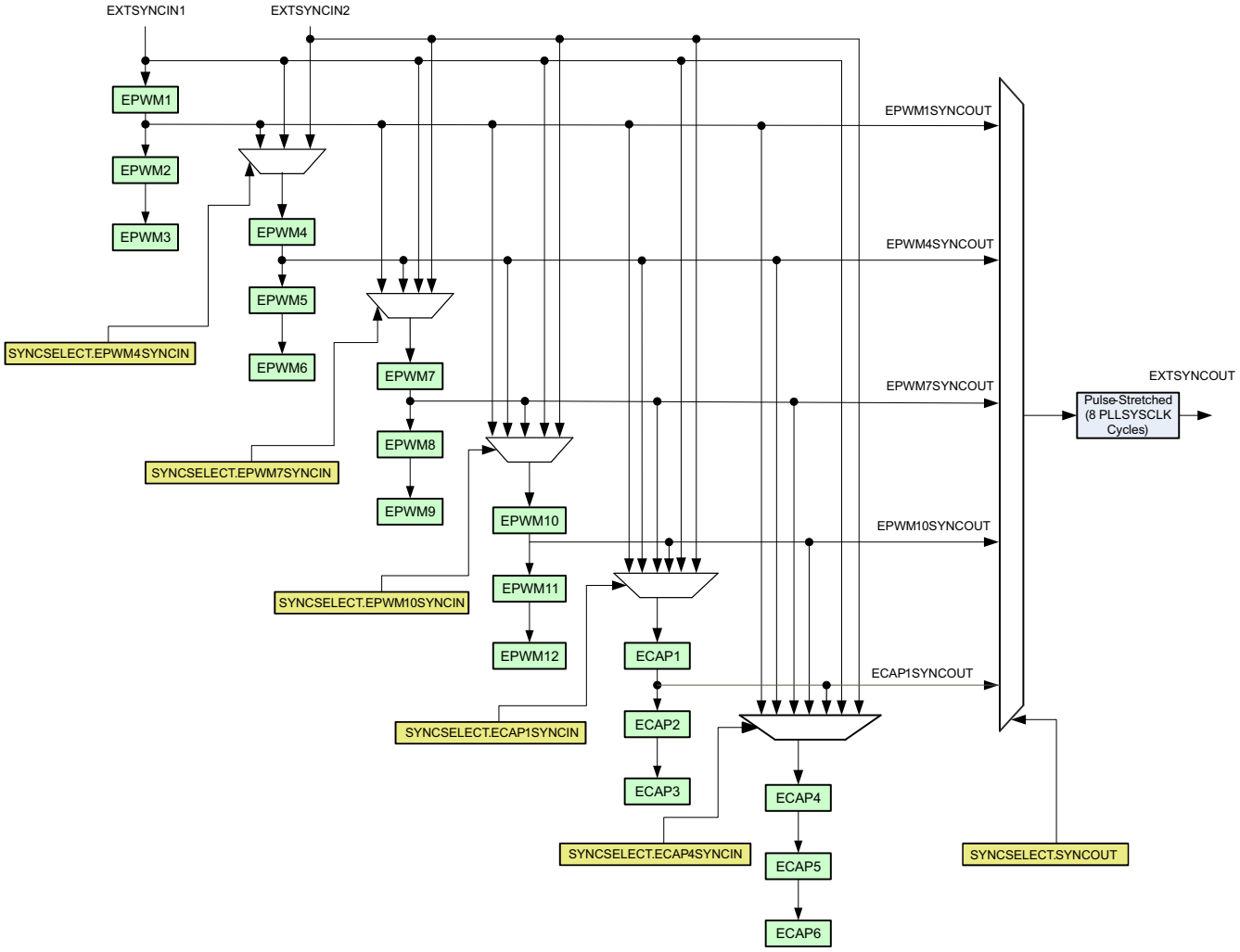



**Figure 15-7. Time-Base Counter Synchronization Scheme**

**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:

- **EPWMxSYNCI: Synchronization Input Pulse:** The value of the phase register is loaded into the counter register when an input synchronization pulse is detected (TBPHS → TBCTR). This operation occurs on the next valid time-base clock (TBCLK) edge.



Copyright © 2024 Texas Instruments Incorporated

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 EPWMxSYNCI.
- **Digital Compare Event Synchronization Pulse:** DCAEVT1 and DCBEVT1 digital compare events can be configured to generate synchronization pulses which have the same affect as EPWMxSYNCI.

**Note**

If the EPWMxSYNCI signal is held high, the sync does not continuously occur. The EPWMxSYNCI 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 15-8 through Figure 15-11 for examples.

Clearing the TBCTL[PHSEN] bit configures the ePWM to ignore the synchronization input pulse. The synchronization pulse can still be allowed to flow-through to the EPWMxSYNCO and be used to synchronize other ePWM modules. In this way, a controller time-base (for example, ePWM1) can be set up and downstream modules (ePWM2 - ePWMx) can run in synchronization with the controller. See Section 15.13 for more details on synchronization strategies.

### *15.4.4 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.

### *15.4.5 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.



Copyright © 2024 Texas Instruments Incorporated

### 15.4.6 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 EPWMxSYNCI signal.

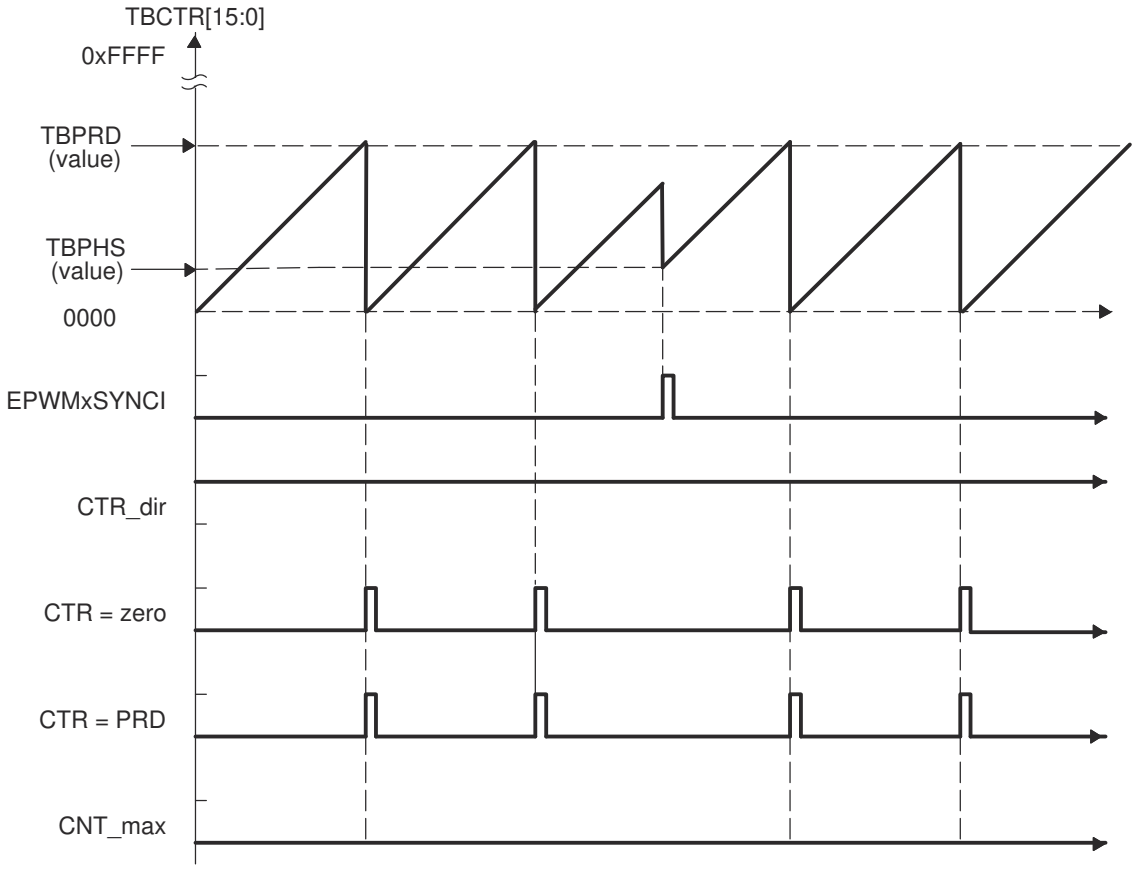



**Figure 15-8. Time-Base Up-Count Mode Waveforms**



Copyright © 2024 Texas Instruments Incorporated

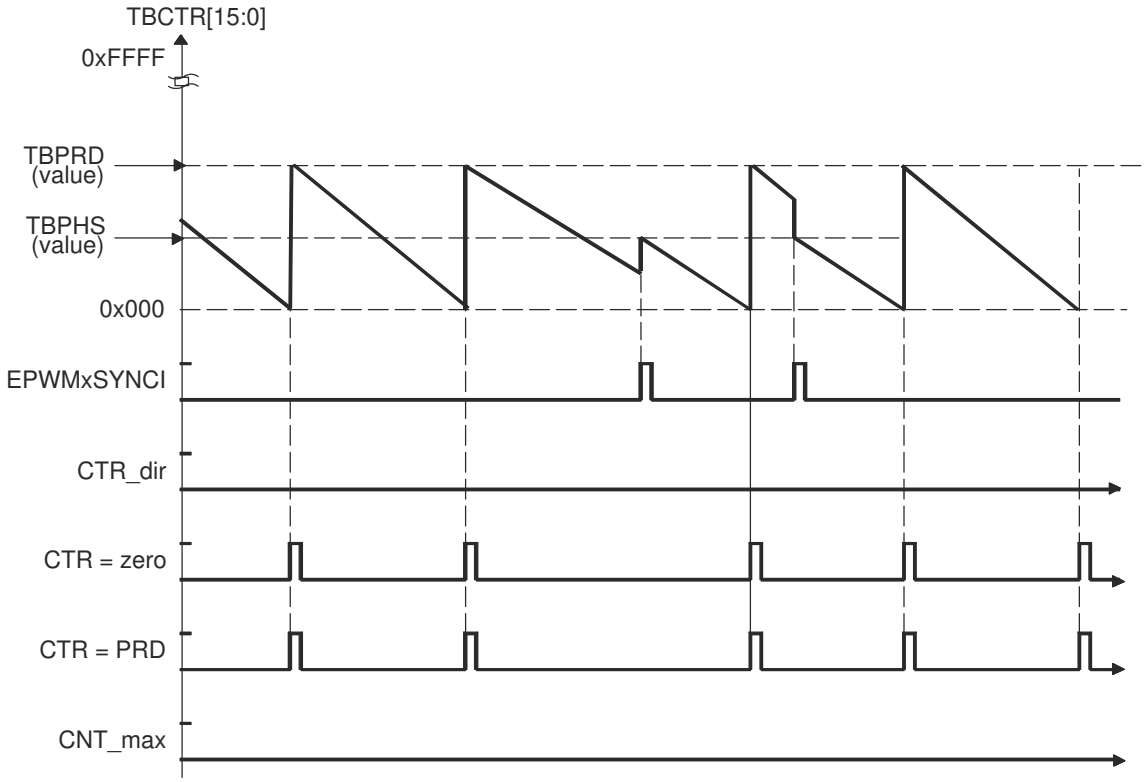



**Figure 15-9. Time-Base Down-Count Mode Waveforms**


Copyright © 2024 Texas Instruments Incorporated

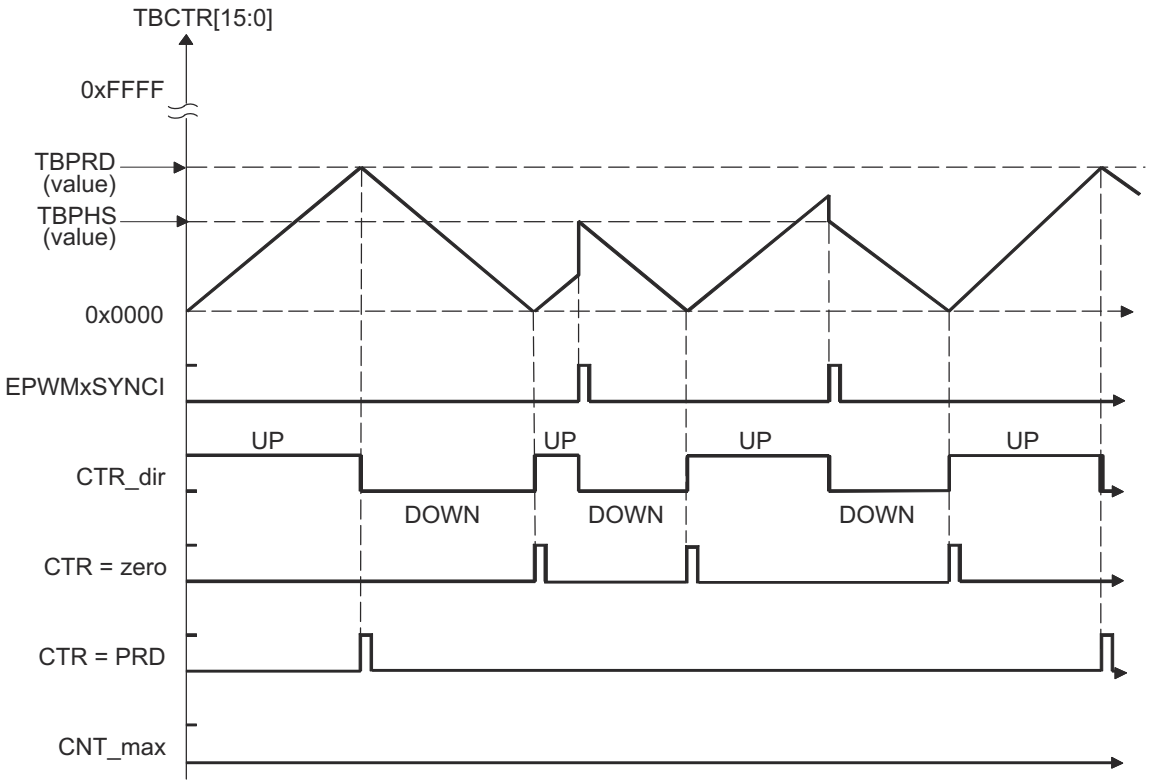



**Figure 15-10. Time-Base Up-Down-Count Waveforms, TBCTL[PHSDIR = 0] Count Down On Synchronization Event**



Copyright © 2024 Texas Instruments Incorporated

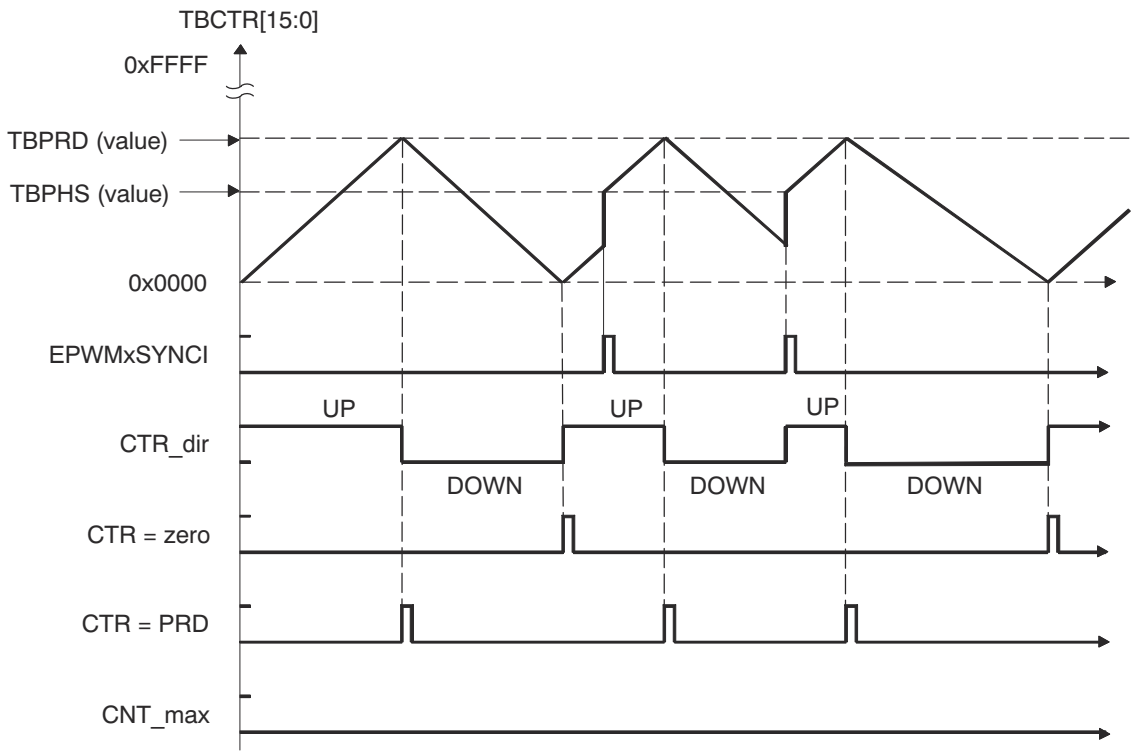



**Figure 15-11. Time-Base Up-Down Count Waveforms, TBCTL[PHSDIR = 1] Count Up On Synchronization Event**



Copyright © 2024 Texas Instruments Incorporated

### 15.4.7 Global Load

Figure 15-12 shows the signals and registers associated with the global load feature.

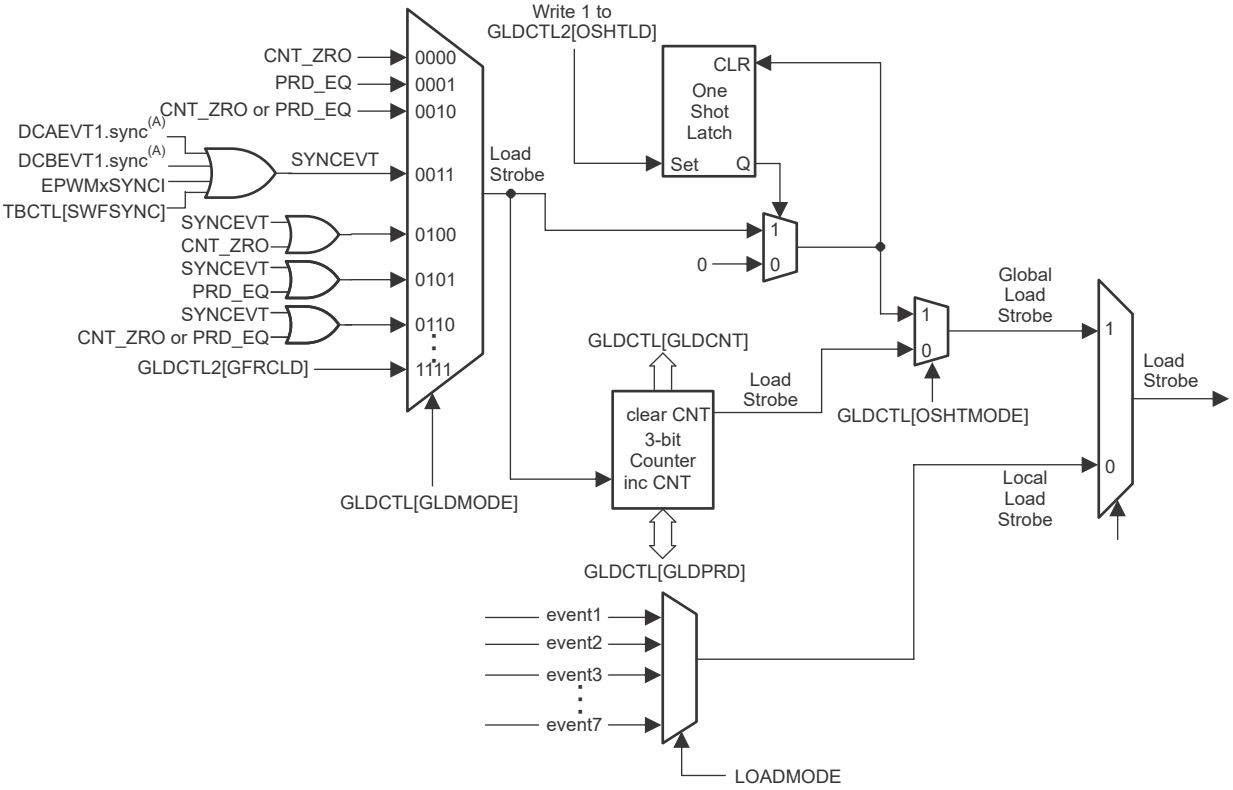



**Figure 15-12. Global Load: Signals and Registers**

**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.

#### 15.4.7.1 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).



Copyright © 2024 Texas Instruments Incorporated

#### 15.4.7.2 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.

#### 15.4.7.3 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 15-13.

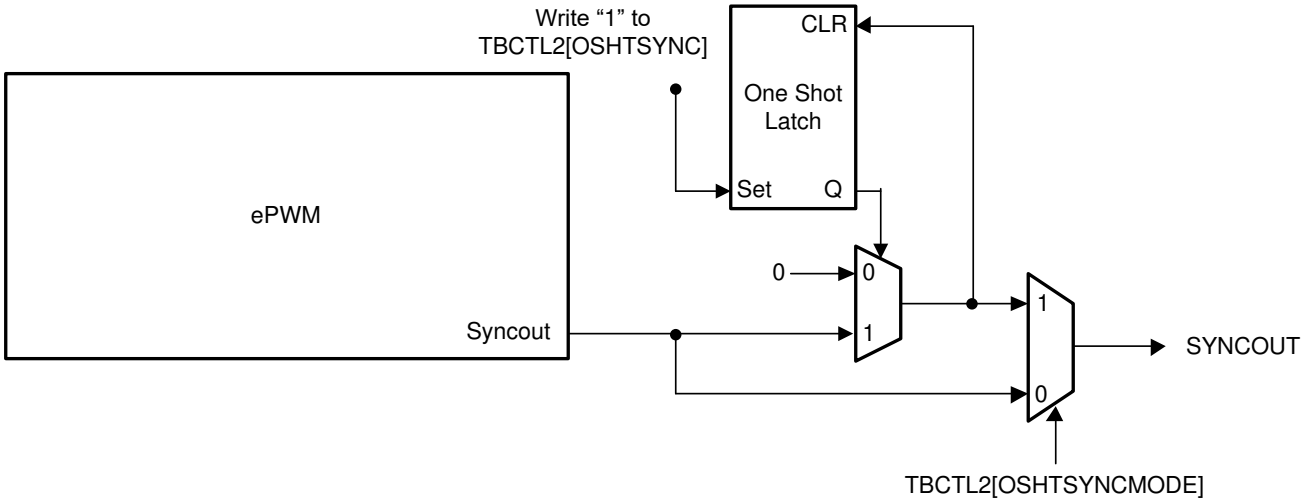



**Figure 15-13. One-Shot Sync Mode**



Copyright © 2024 Texas Instruments Incorporated

## 15.5 Counter-Compare (CC) Submodule

Figure 15-14 illustrates the counter-compare submodule within the ePWM.

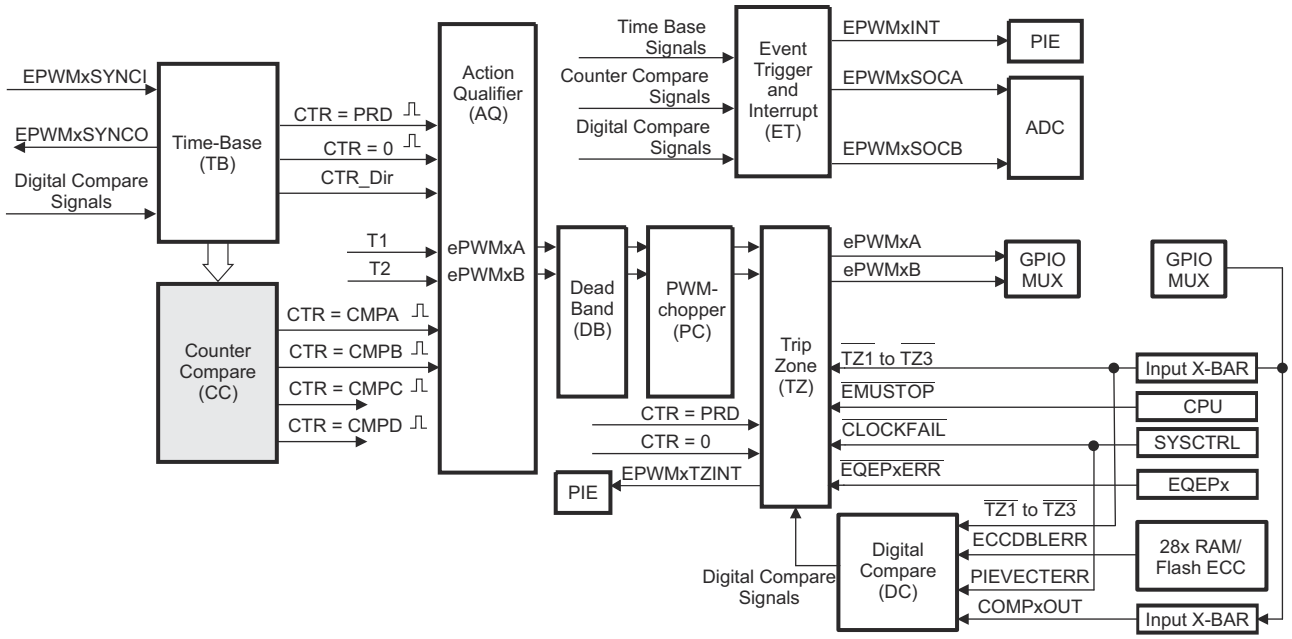



**Figure 15-14. Counter-Compare Submodule**

### *15.5.1 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:

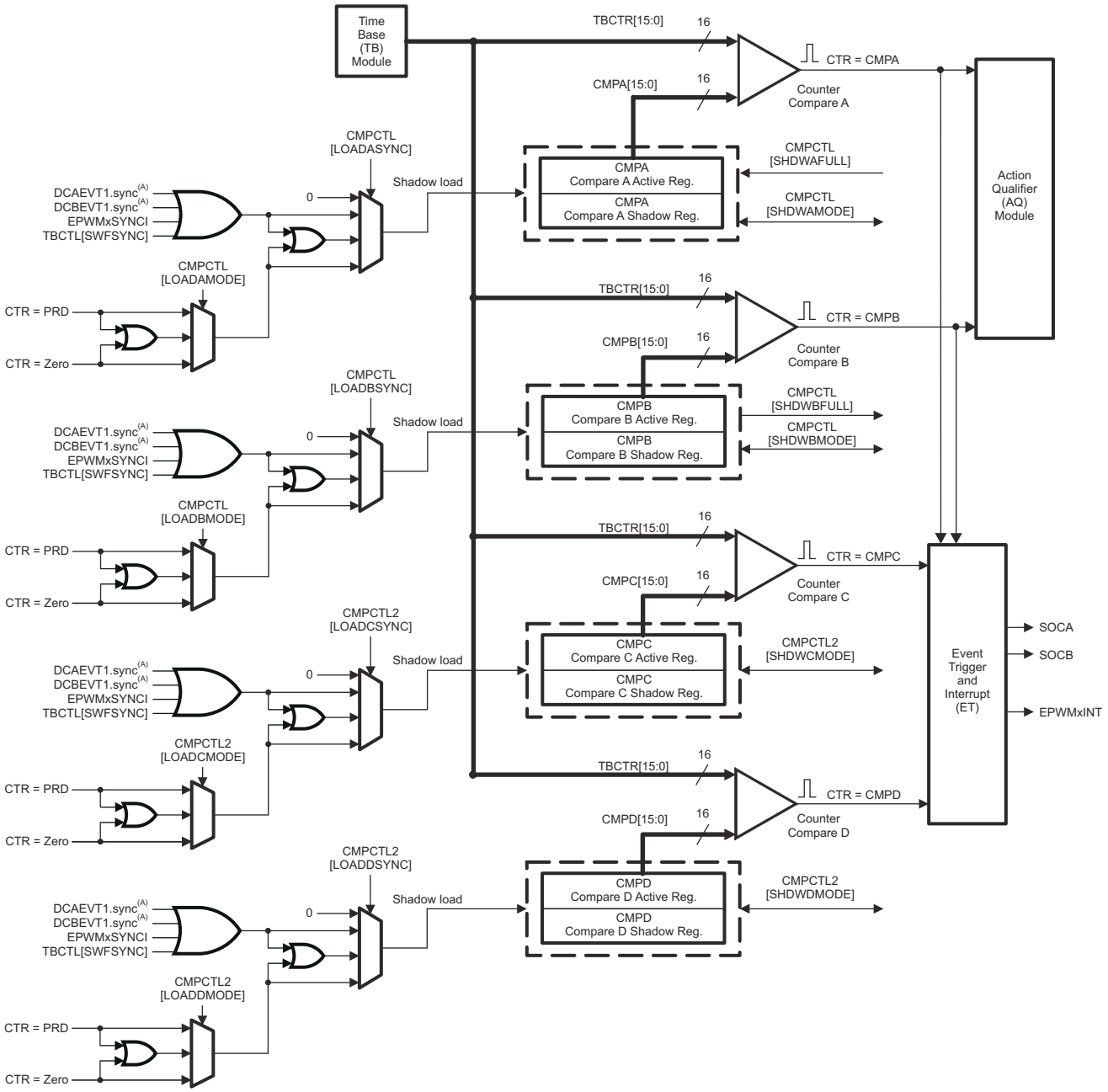
- Generates events based on programmable time stamps using the CMPA, CMPB, CMPC, and CMPD registers:
  - CTR = CMPA: Time-base counter equals counter-compare A register (TBCTR = CMPA)
  - CTR = CMPB: Time-base counter equals counter-compare B register (TBCTR = CMPB)
  - CTR = CMPC: Time-base counter 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



Copyright © 2024 Texas Instruments Incorporated

### 15.5.2 Controlling and Monitoring the Counter-Compare Submodule

The counter-compare submodule operation is shown in Figure 15-15.




A. These events are generated by the ePWM digital compare (DC) submodule based on the levels of the TRIPIN inputs (for example, CMPSSx and $\overline{TZ}$ signals).

**Figure 15-15. Detailed View of the Counter-Compare Submodule**



Copyright © 2024 Texas Instruments Incorporated

### 15.5.3 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 applies to the action-qualifier submodule where the events are qualified by the counter direction and converted into actions if enabled. Refer to Section 15.6.1 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 EPWMxSYNCI 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 15.6.5 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.



Copyright © 2024 Texas Instruments Incorporated

**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 EPWMxSYNCI 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 15.4.7.

### *15.5.4 Count Mode Timing Waveforms*

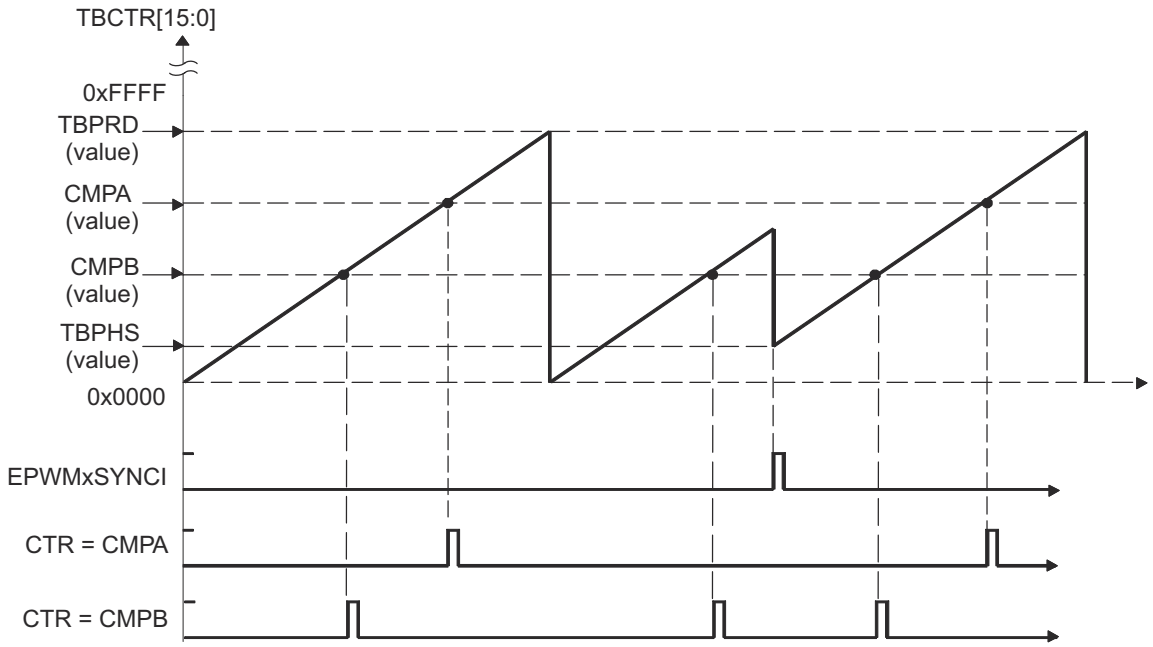
The counter-compare module can generate compare events in all three count modes:

- 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 15-16 through Figure 15-19 show when events are generated and how the EPWMxSYNCI signal interacts.



Copyright © 2024 Texas Instruments Incorporated




An EPWMxSYNCI 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 15-16. Counter-Compare Event Waveforms in Up-Count Mode**

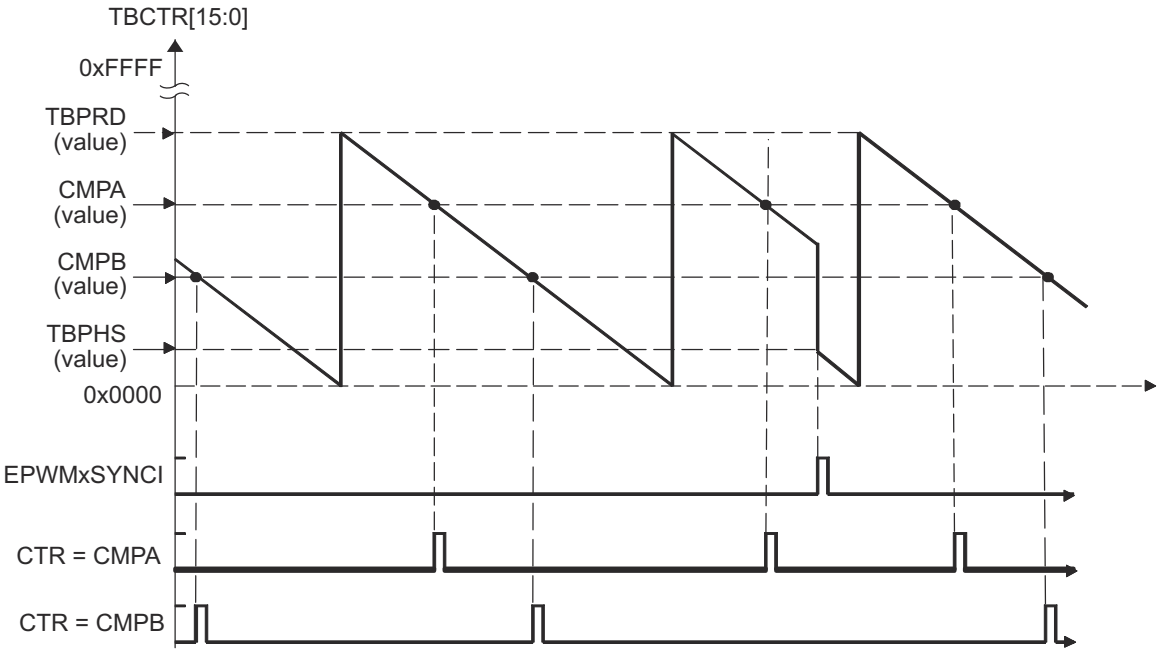



**Figure 15-17. Counter-Compare Events in Down-Count Mode**



Copyright © 2024 Texas Instruments Incorporated

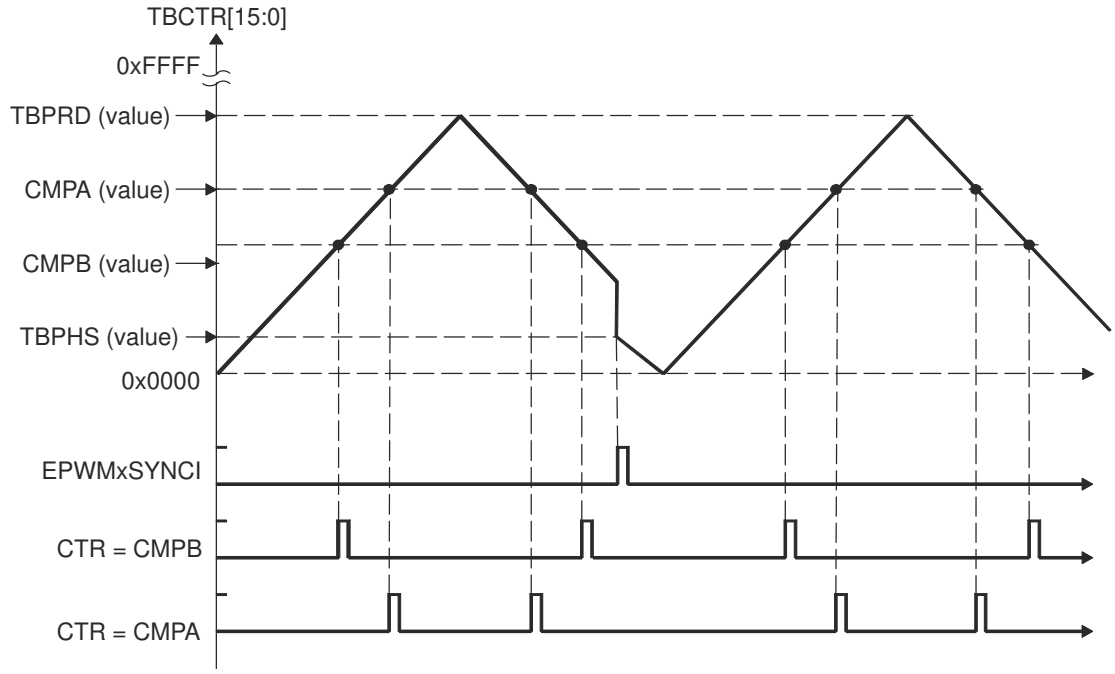



**Figure 15-18. Counter-Compare Events In Up-Down-Count Mode, TBCTL[PHSDIR = 0] Count Down On Synchronization Event**

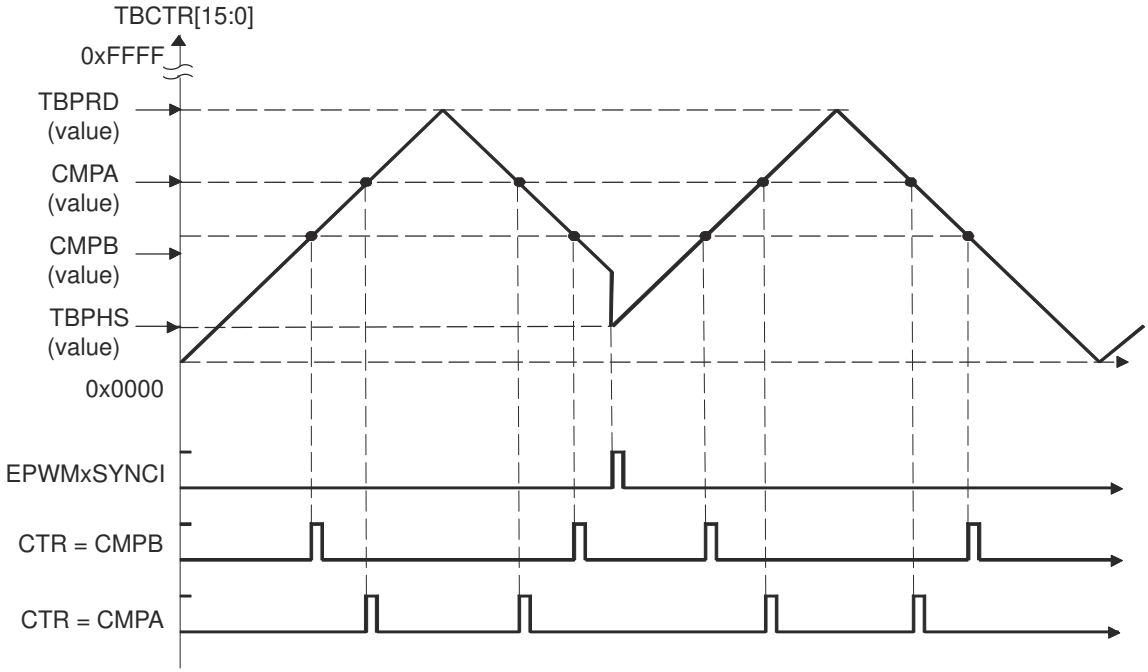



**Figure 15-19. Counter-Compare Events In Up-Down-Count Mode, TBCTL[PHSDIR = 1] Count Up On Synchronization Event**


Copyright © 2024 Texas Instruments Incorporated

## 15.6 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 15-20 illustrates the action-qualifier submodule within the ePWM.

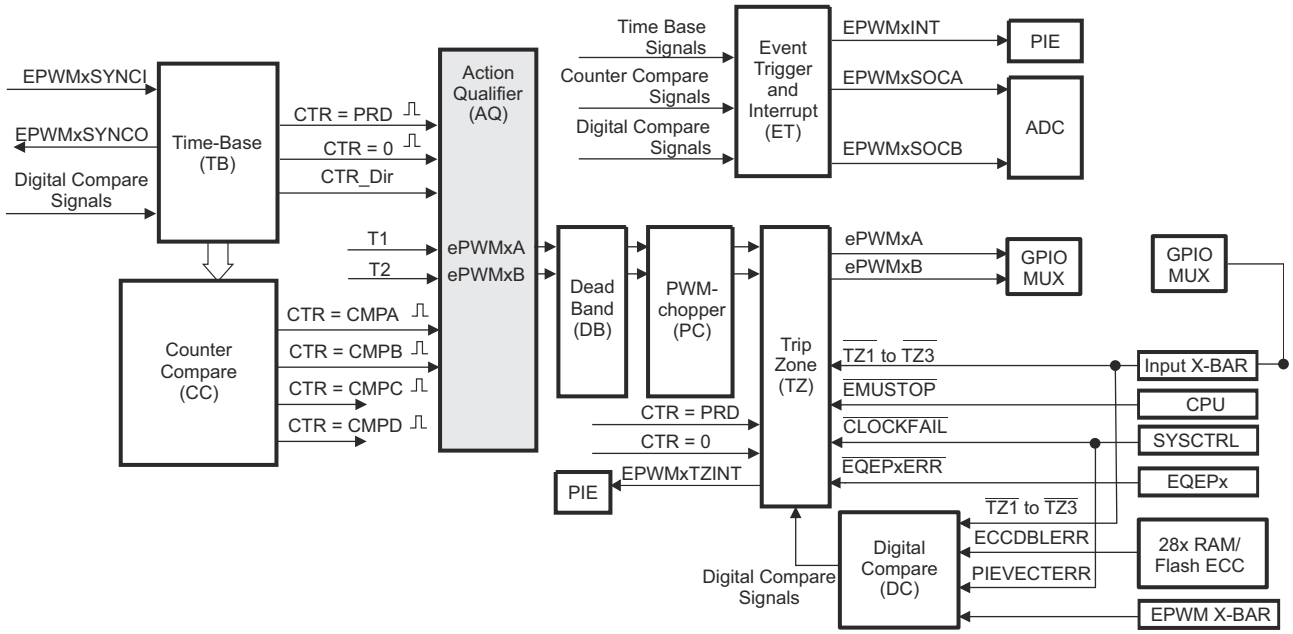



**Figure 15-20. Action-Qualifier Submodule**

### *15.6.1 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



Copyright © 2024 Texas Instruments Incorporated

### 15.6.2 Action-Qualifier Submodule Control and Status Register Definitions

The action-qualifier submodule operation is shown in Figure 15-21 and monitored by way of the registers in Section 15.15.

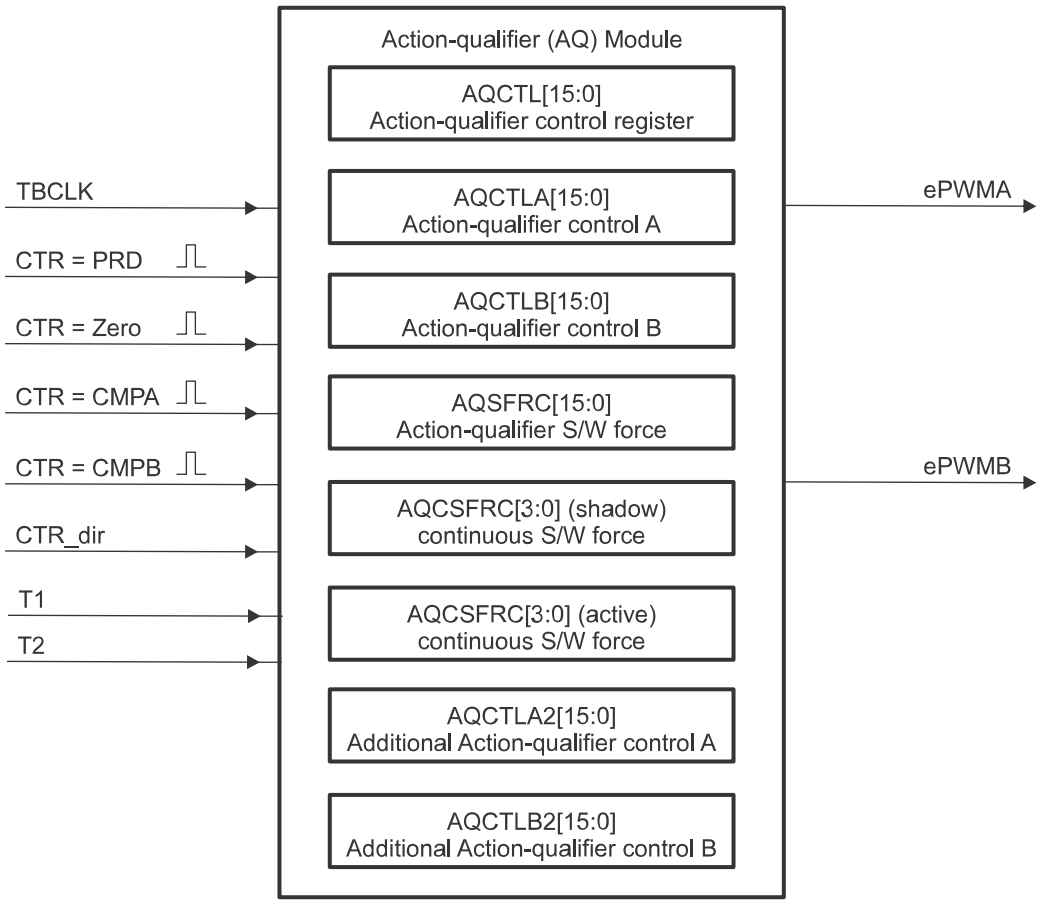



**Figure 15-21. Action-Qualifier Submodule Inputs and Outputs**

For convenience, the possible input events are summarized again in Table 15-3.

**Table 15-3. Action-Qualifier Submodule Possible Input Events**

| Signal | Description | Registers Compared |
|---|---|---|
| CTR = PRD | Time-base counter equal to the period value | TBCTR = TBPRD |
| CTR = Zero | Time-base counter equal to 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.



Copyright © 2024 Texas Instruments Incorporated

**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 15.10 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 15-22. 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).

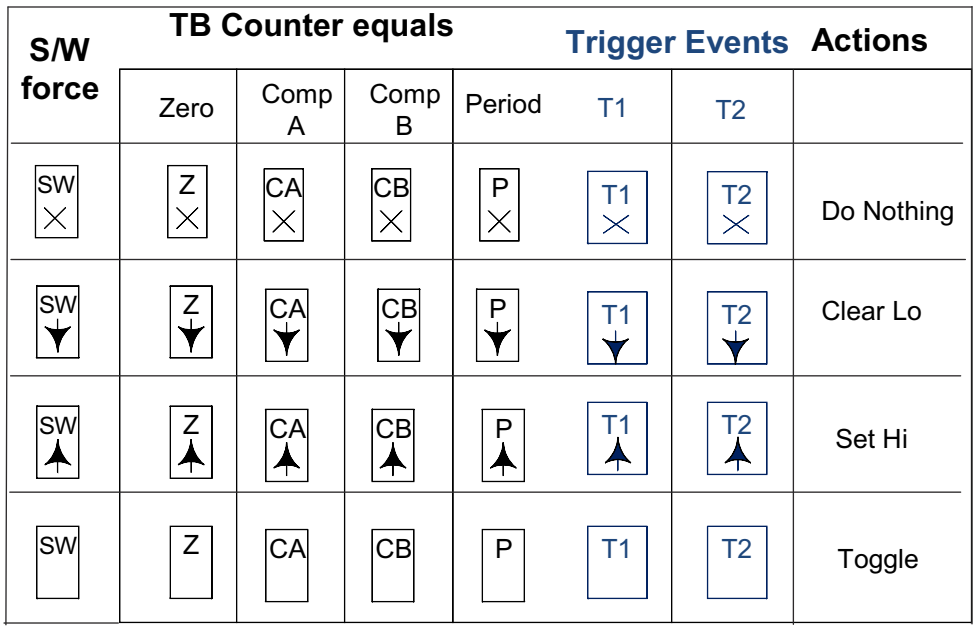



**Figure 15-22. Possible Action-Qualifier Actions for EPWMxA and EPWMxB Outputs**

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



Copyright © 2024 Texas Instruments Incorporated

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.

### 15.6.3 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 15-4. A priority level of 1 is the highest priority and level 10 is the lowest. The priority changes slightly depending on the direction of TBCTR.

**Table 15-4. Action-Qualifier Event Priority for Up-Down-Count Mode**

| Priority Level | Event If TBCTR is Incrementing TBCTR = Zero up to TBCTR = TBPRD | Event If TBCTR is Decrementing TBCTR = TBPRD down to TBCTR = 1 |
|---|---|---|
| 1 (Highest) | Software forced event | Software forced event |
| 2 | T1 on up-count (T1U) | T1 on down-count (T1D) |
| 3 | T2 on up-count (T2U) | T2 on down-count (T2D) |
| 4 | Counter equals CMPB on up-count (CBU) | Counter equals CMPB on down-count (CBD) |
| 5 | Counter equals CMPA on up-count (CAU) | Counter equals CMPA on down-count (CAD) |
| 6 (Lowest) | Counter equals zero | Counter equals period (TBPRD) |

Table 15-5 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 15-5. Action-Qualifier Event Priority for Up-Count Mode**

| Priority Level | Event |
|---|---|
| 1 (Highest) | Software forced event |
| 2 | Counter equal to period (TBPRD) |
| 3 | T1 on up-count (T1U) |
| 4 | T2 on up-count (T2U) |
| 5 | Counter equal to CMPB on up-count (CBU) |
| 6 | Counter equal to CMPA on up-count (CAU) |
| 7 (Lowest) | Counter equal to Zero |

Table 15-6 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 15-6. Action-Qualifier Event Priority for Down-Count Mode**

| Priority Level | Event |
|---|---|
| 1 (Highest) | Software forced event |
| 2 | Counter equal to Zero |
| 3 | T1 on down-count (T1D) |
| 4 | T2 on down-count (T2D) |
| 5 | Counter equal to CMPB on down-count (CBD) |
| 6 | Counter equal to CMPA on down-count (CAD) |
| 7 (Lowest) | Counter equal to period (TBPRD) |



Copyright © 2024 Texas Instruments Incorporated

It is possible to set the compare value greater than the period. In this case, the action takes place as shown in Table 15-7.

**Table 15-7. Behavior if CMPA/CMPB is Greater than the Period**

| Counter Mode | Compare on Up-Count Event CAD/CBD | Compare on Down-Count Event CAD/CBD |
|---|---|---|
| Up-Count Mode | If CMPA/CMPB ≤ TBPRD period, then the event occurs on a compare match (TBCTR=CMPA or CMPB). | Never occurs. |
| | If CMPA/CMPB > TBPRD, then the event does not occur. | |
| Down-Count Mode | Never occurs. | If CMPA/CMPB < TBPRD, the event occurs on a compare match (TBCTR=CMPA or CMPB). |
| | | If CMPA/CMPB ≥ 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 < TBPRD and the counter is decrementing, the event occurs on a compare match (TBCTR=CMPA or CMPB). |
| | If CMPA/CMPB is ≥ TBPRD, the event occurs on a period match (TBCTR = TBPRD). | If CMPA/CMPB ≥ TBPRD, the event occurs on a period match (TBCTR=TBPRD). |

### *15.6.4 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 EPWMxSYNCI 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 15.4.7.

**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 15-23 and Figure 15-24.

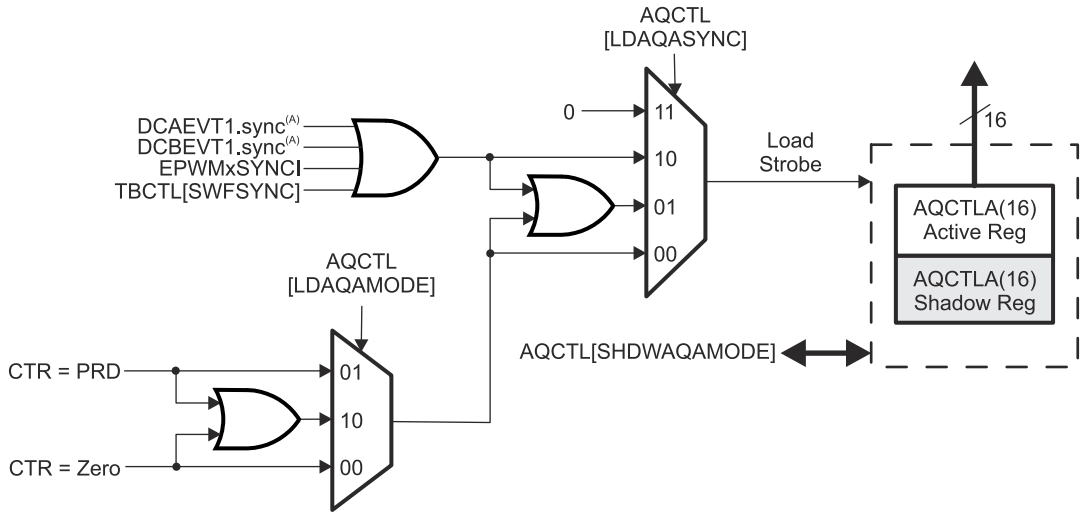


Copyright © 2024 Texas Instruments Incorporated

**Note**

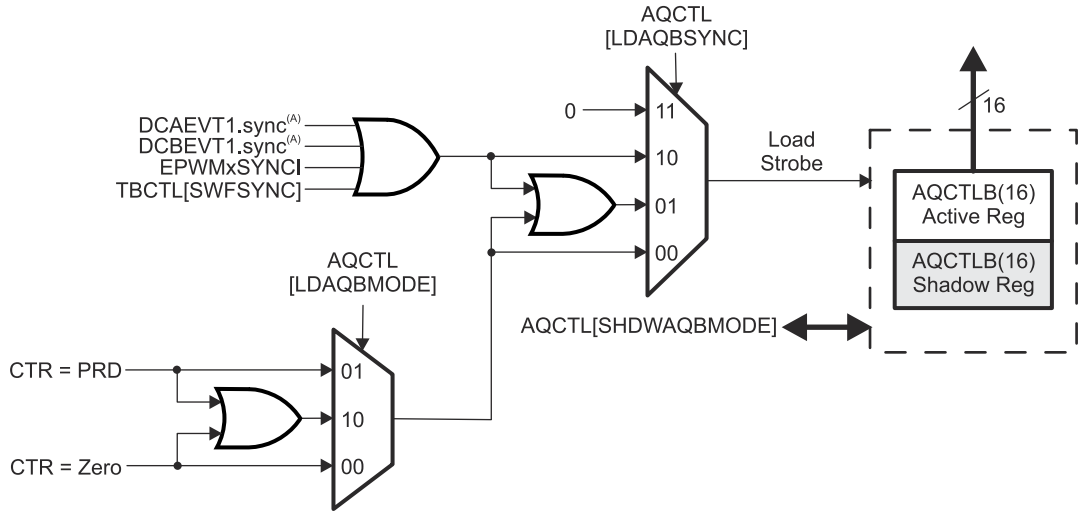
Shadow to Active Load of Action Qualifier Output A/B Control Register [AQCTLA and AQCTLB] on CMPA = 0 or CMPB = 0 boundary

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 $\overline{TZ}$ signals).

**Figure 15-23. 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 $\overline{TZ}$ signals).

**Figure 15-24. AQCTL[SHDWAQBMODE]**



Copyright © 2024 Texas Instruments Incorporated

### 15.6.5 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 < Deadband) or (CMPX > PRD – 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 < Deadband/2) or (CMPX > PRD – (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 15-25 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.



Copyright © 2024 Texas Instruments Incorporated

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.

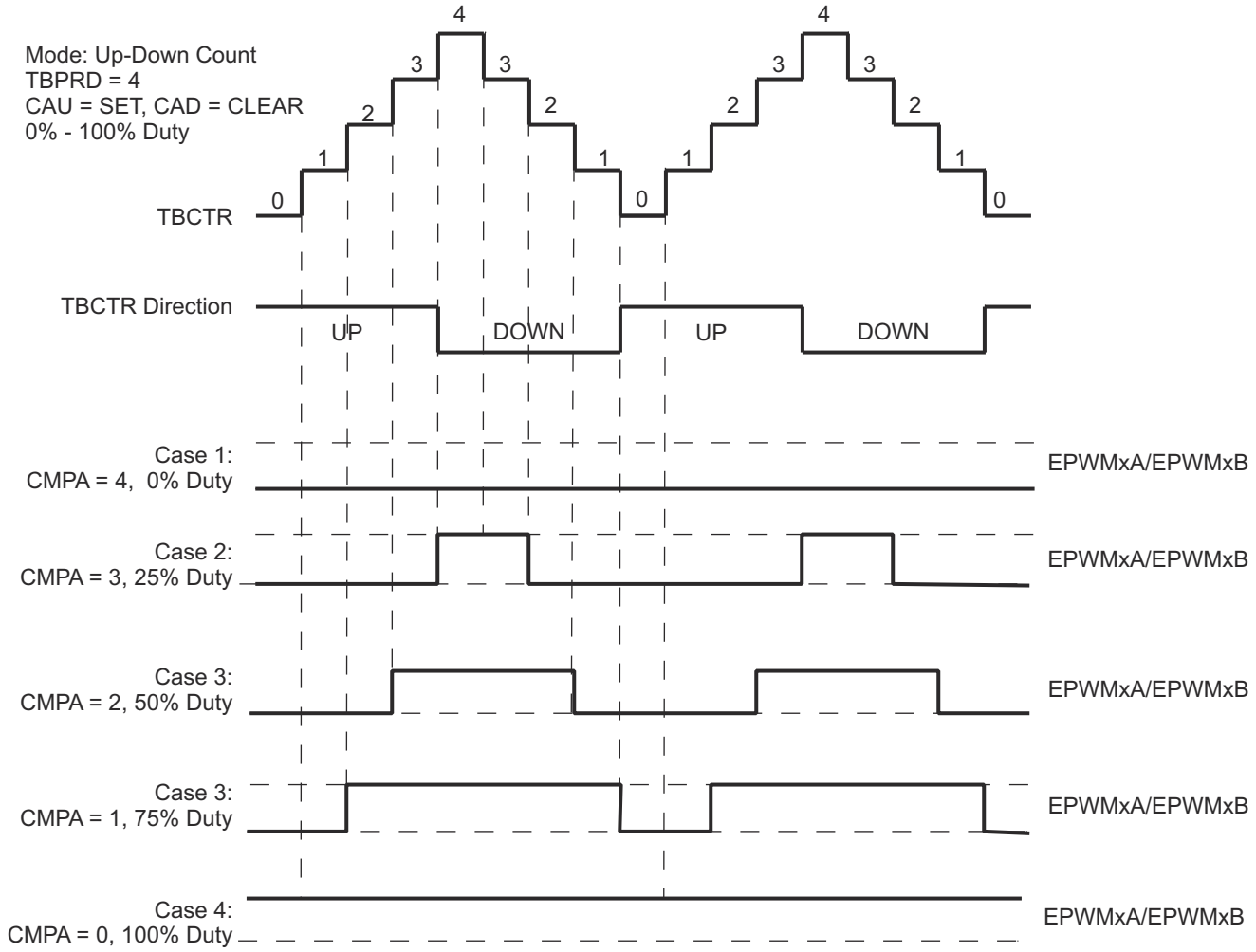


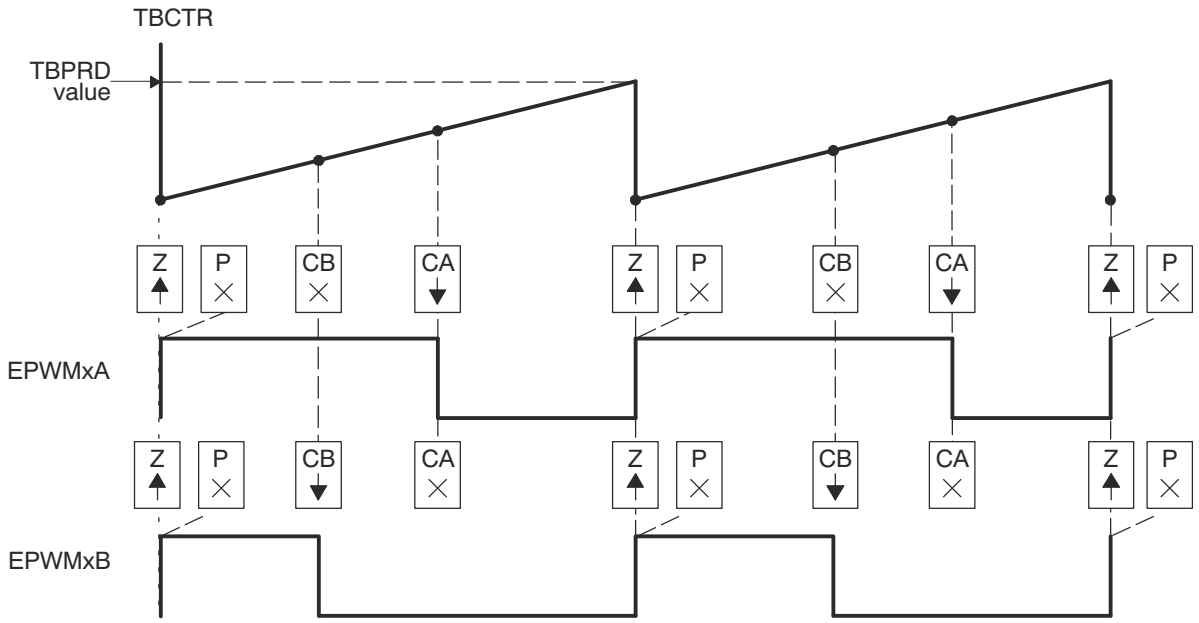

**Figure 15-25. Up-Down Count Mode Symmetrical Waveform**

The PWM waveforms in Figure 15-26 through Figure 15-31 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



Copyright © 2024 Texas Instruments Incorporated

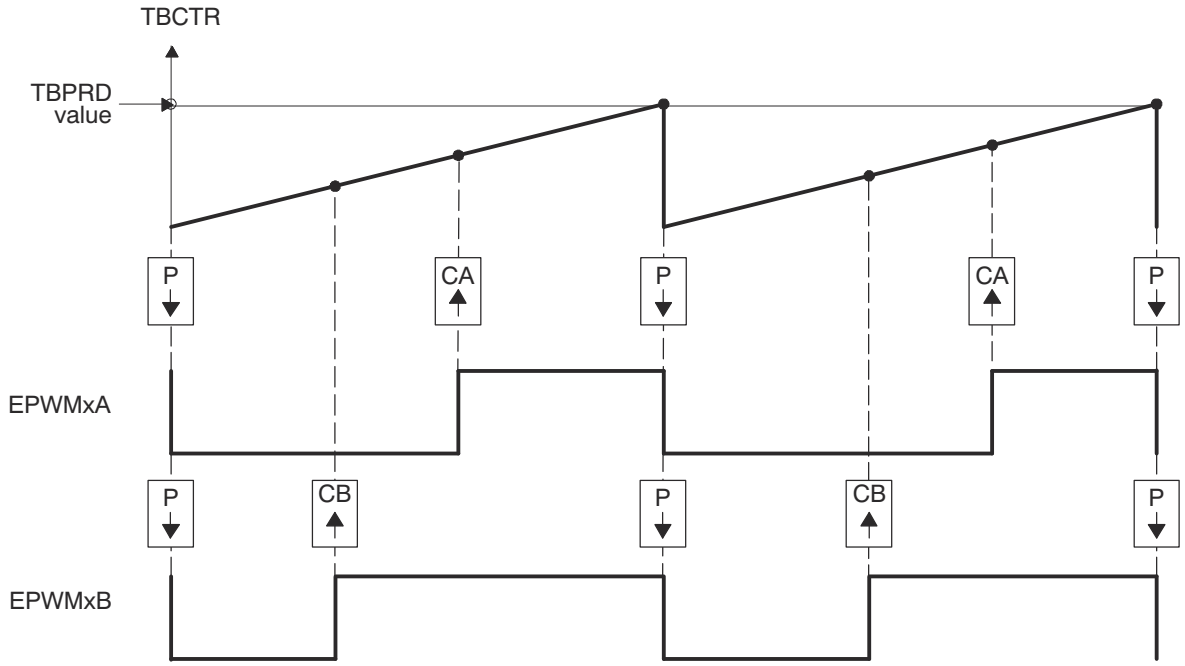



A. PWM period = (TBPRD + 1) × $T_{TBCLK}$
B. Duty modulation for EPWMxA is set by CMPA, and is active high (that is, high time duty proportional to CMPA).
C. Duty modulation for EPWMxB is set by CMPB and is active high (that is, high time duty proportional to CMPB).
D. The "Do Nothing" actions (X) are shown for completeness, but are not shown on subsequent diagrams.
E. Actions at zero and period, although appearing to occur concurrently, are actually separated by one TBCLK period. TBCTR wraps from period to 0000.

**Figure 15-26. Up, Single Edge Asymmetric Waveform, with Independent Modulation on EPWMxA and EPWMxB—Active High**



Copyright © 2024 Texas Instruments Incorporated

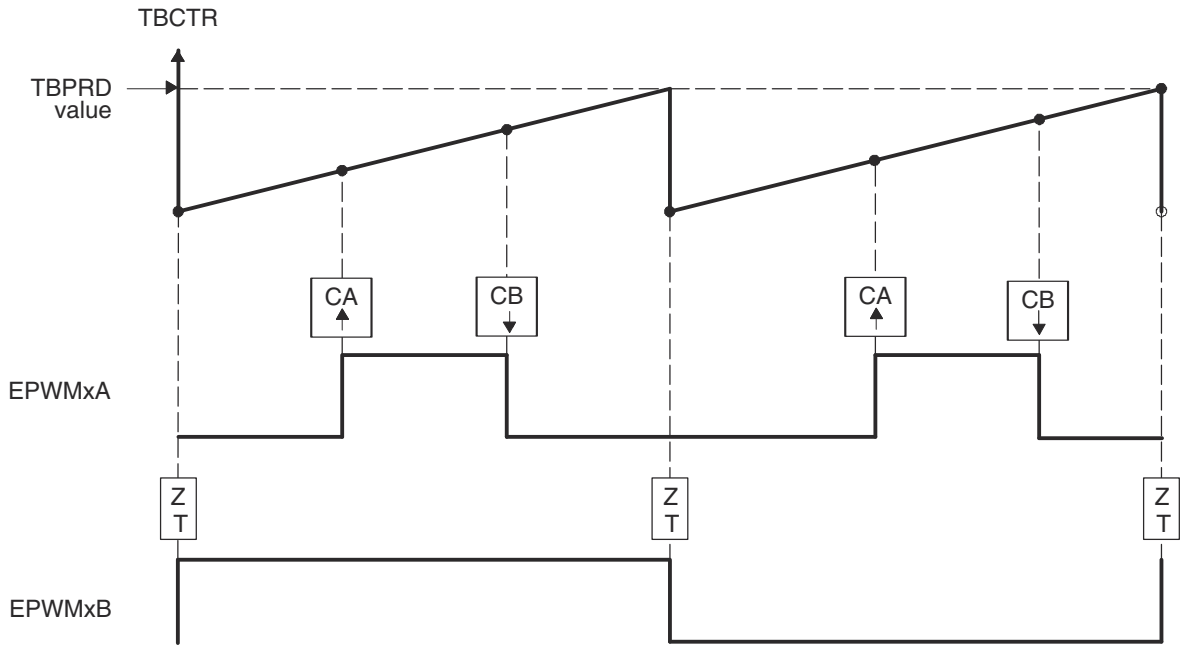



A. PWM period = (TBPRD + 1) × $T_{TBCLK}$
B. Duty modulation for EPWMxA is set by CMPA, and is active low (that is, the low time duty is proportional to CMPA).
C. Duty modulation for EPWMxB is set by CMPB and is active low (that is, the low time duty is proportional to CMPB).
D. Actions at zero and period, although appearing to occur concurrently, are actually separated by one TBCLK period. TBCTR wraps from period to 0000.

**Figure 15-27. Up, Single Edge Asymmetric Waveform with Independent Modulation on EPWMxA and EPWMxB—Active Low**

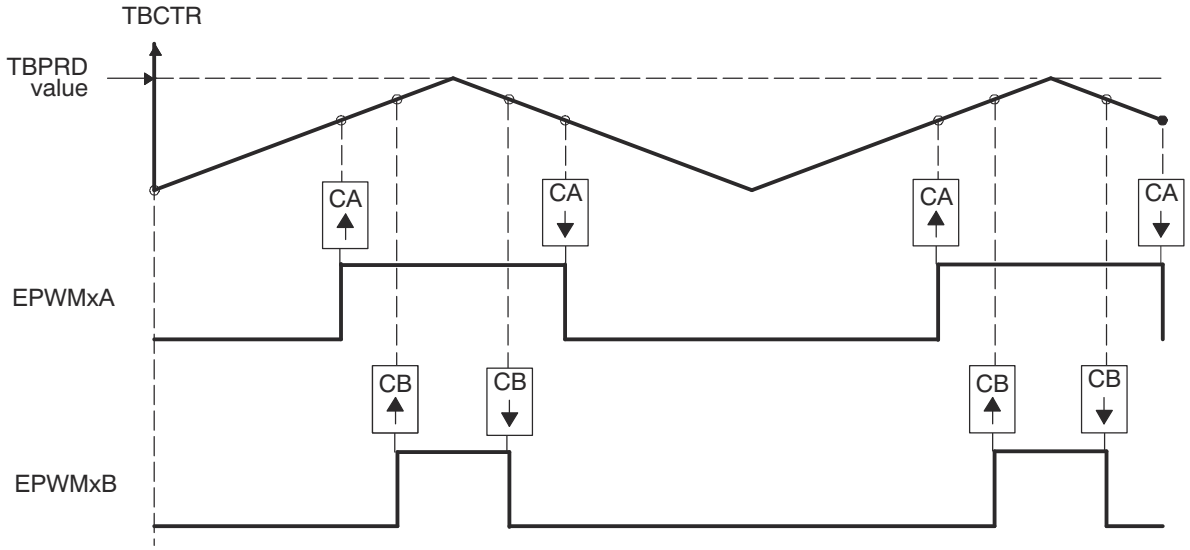


Copyright © 2024 Texas Instruments Incorporated




A. PWM frequency = 1/((TBPRD + 1) × $T_{TBCLK}$)
B. Pulse can be placed anywhere within the PWM cycle (0000 - TBPRD)
C. High time duty proportional to (CMPB - CMPA)

**Figure 15-28. Up-Count, Pulse Placement Asymmetric Waveform With Independent Modulation on EPWMxA**

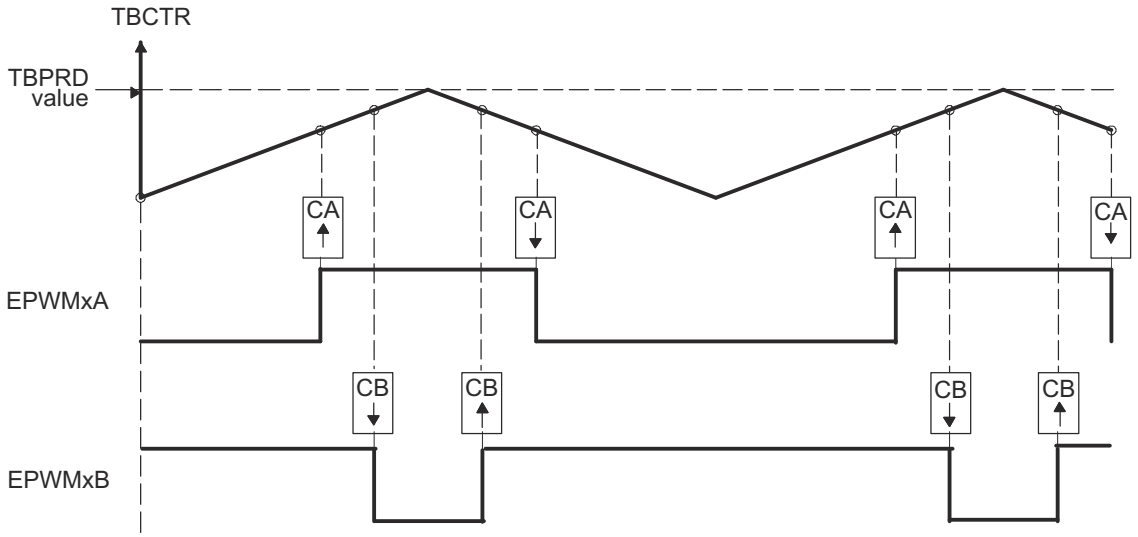



A. PWM period = 2 x TBPRD × $T_{TBCLK}$
B. Duty modulation for EPWMxA is set by CMPA, and is active low (that is, the low time duty is proportional to CMPA).
C. Duty modulation for EPWMxB is set by CMPB and is active low (that is, the low time duty is proportional to CMPB).
D. Outputs EPWMxA and EPWMxB can drive independent power switches.

**Figure 15-29. Up-Down Count, Dual-Edge Symmetric Waveform, with Independent Modulation on EPWMxA and EPWMxB — Active Low**

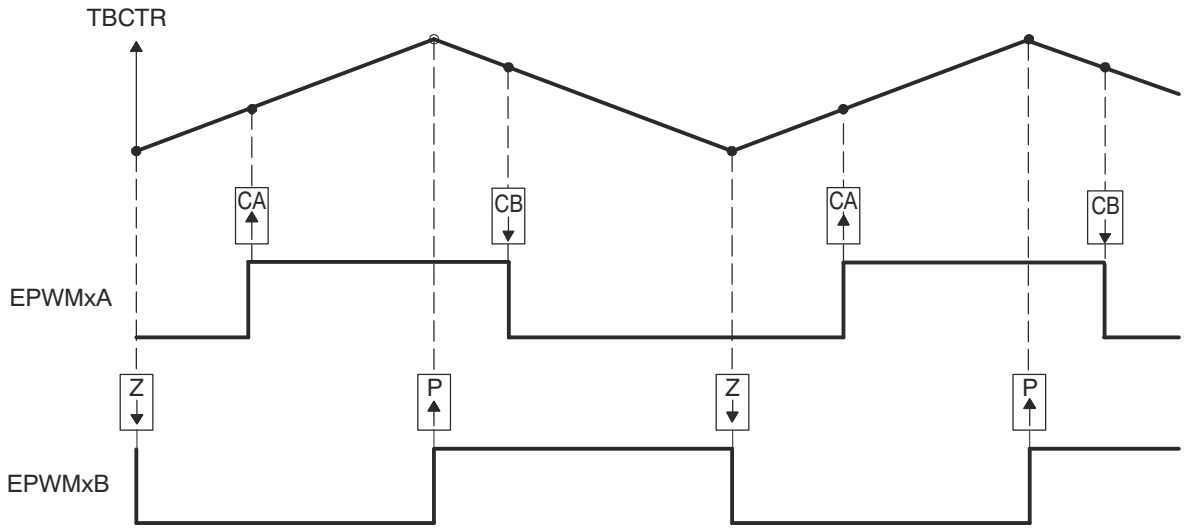


Copyright © 2024 Texas Instruments Incorporated




A. PWM period = 2 × TBPRD × $T_{TBCLK}$
B. Duty modulation for EPWMxA is set by CMPA, and is active low, that is, low time duty proportional to CMPA.
C. Duty modulation for EPWMxB is set by CMPB and is active high, that is, high time duty proportional to CMPB.
D. Outputs EPWMx can drive upper/lower (complementary) power switches.
E. Dead-band = CMPB - CMPA (fully programmable edge placement by software). Note the dead-band module is also available if the more classical edge delay method is required.

**Figure 15-30. Up-Down Count, Dual-Edge Symmetric Waveform, with Independent Modulation on EPWMxA and EPWMxB — Complementary**

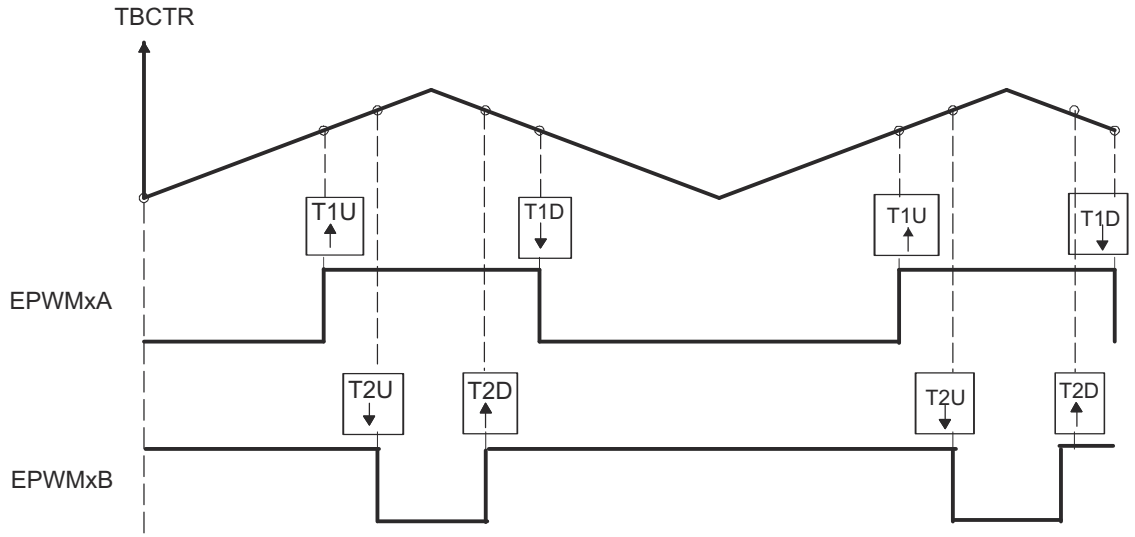



A. PWM period = 2 × TBPRD × TBCLK
B. Rising edge and falling edge can be asymmetrically positioned within a PWM cycle. This allows for pulse placement techniques.
C. Duty modulation for EPWMxA is set by CMPA and CMPB.
D. Low time duty for EPWMxA is proportional to (CMPA + CMPB).
E. To change this example to active high, CMPA and CMPB actions need to be inverted (that is, Clear on CMPA, Set on CMPB).
F. Duty modulation for EPWMxB is fixed at 50% (utilizes spare action resources for EPWMxB).

**Figure 15-31. Up-Down Count, Dual-Edge Asymmetric Waveform, with Independent Modulation on EPWMxA—Active Low**



Copyright © 2024 Texas Instruments Incorporated




A. PWM period = 2 × TBPRD × 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 15-32. Up-Down Count, PWM Waveform Generation Utilizing T1 and T2 Events**



Copyright © 2024 Texas Instruments Incorporated

## 15.7 Dead-Band Generator (DB) Submodule

Figure 15-33 illustrates the dead-band submodule within the ePWM.

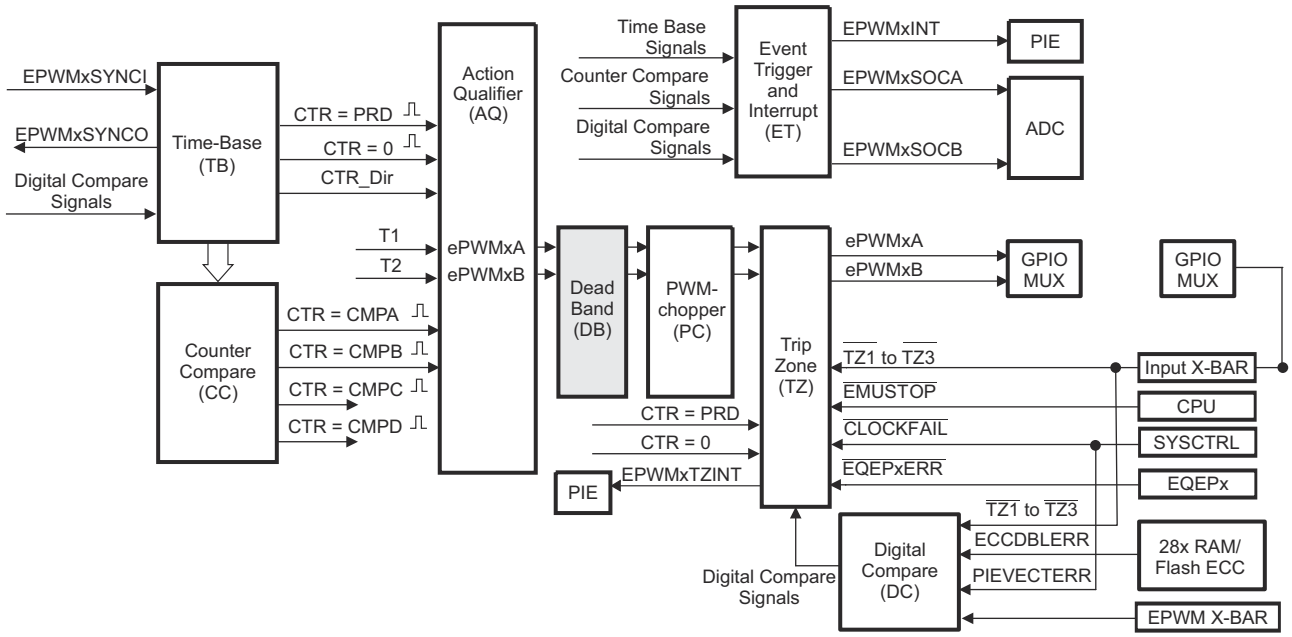



**Figure 15-33. Dead_Band Submodule**

### *15.7.1 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)



Copyright © 2024 Texas Instruments Incorporated

### 15.7.2 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 15-34, 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.



Copyright © 2024 Texas Instruments Incorporated

**Shadow Mode:**

The shadow mode for the DBRED is enabled by setting the DBCTL[SHDWDBREDMODE] bit and the shadow register for DBFED is enabled by setting the DBCTL [SHDWDBFEDMODE] bit. Shadow mode is disabled by default for both DBRED and DBFED

If the shadow register is enabled, then the content of the shadow register is transferred to the active register on one of the following events as specified by the DBCTL [LOADREDMODE] 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[SHDWDBREDMODE] 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 15.4.7.

**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.



Copyright © 2024 Texas Instruments Incorporated

### 15.7.3 Operational Highlights for the Dead-Band Submodule

The configuration options for the dead-band submodule are shown in Figure 15-34.

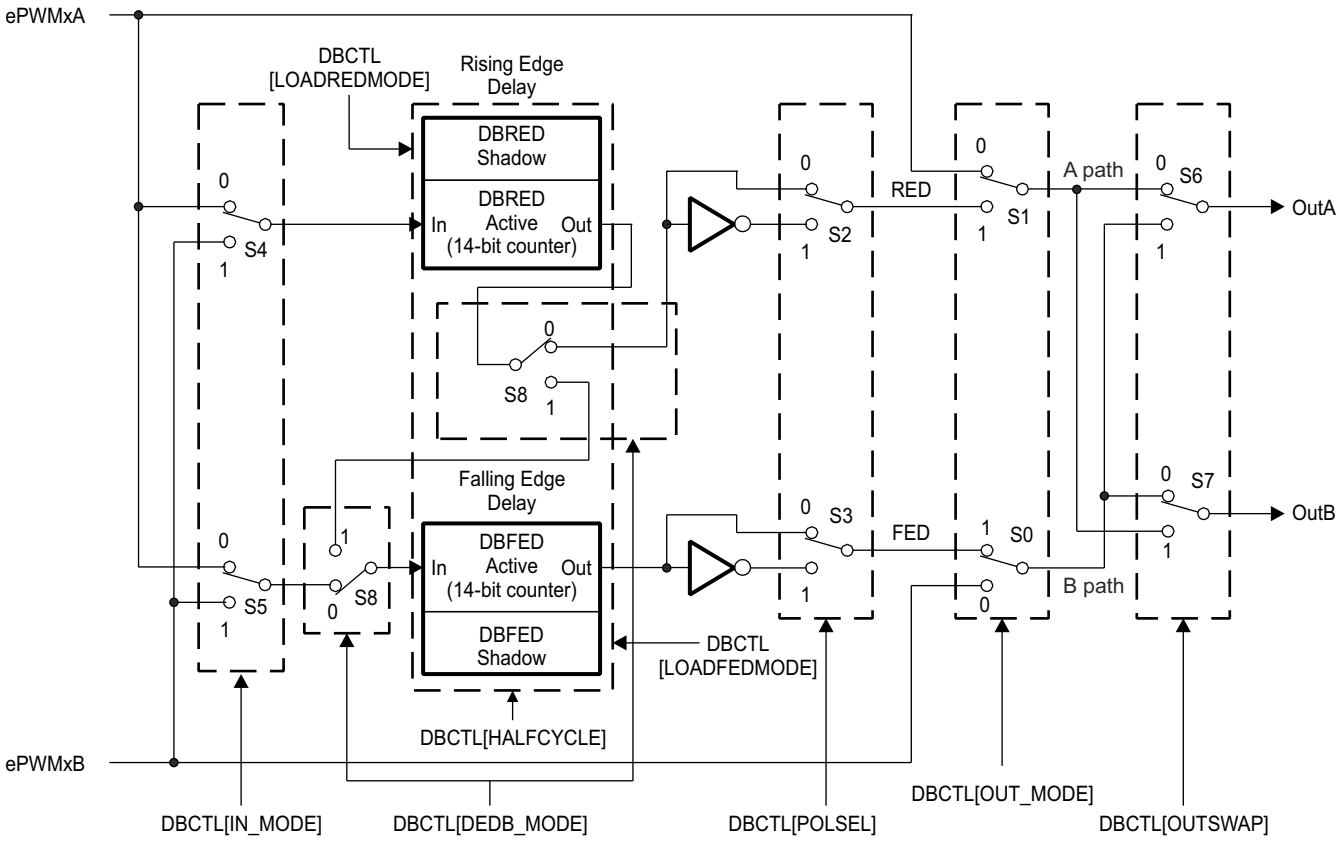



**Figure 15-34. Configuration Options for the Dead-Band Submodule**

Although all combinations are supported, not all are typical usage modes. Table 15-8 documents some classical dead-band configurations. These modes assume that the DBCTL[IN_MODE] is configured such that EPWMxA In is the source for both falling-edge and rising-edge delay. Enhanced, or non-traditional modes can be achieved by changing the input signal source. The modes shown in Table 15-8 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 15-35. Note that to generate equivalent waveforms to Figure 15-35, 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 15-8 show combinations where either the falling-edge delay (FED) or rising-edge delay (RED) blocks are bypassed.

Figure 15-35 shows waveforms for typical cases where 0% < duty < 100%.



Copyright © 2024 Texas Instruments Incorporated

**Table 15-8. Classical Dead-Band Operating Modes**

| Mode | Mode Description | DBCTL[POLSEL] | | DBCTL[OUT_MODE] | |
|---|---|---|---|---|---|
| | | S3 | S2 | S1 | S0 |
| 1 | EPWMxA and EPWMxB Passed Through (No Delay) | X | X | 0 | 0 |
| 2 | Active High Complementary (AHC) | 1 | 0 | 1 | 1 |
| 3 | Active Low Complementary (ALC) | 0 | 1 | 1 | 1 |
| 4 | Active High (AH) | 0 | 0 | 1 | 1 |
| 5 | Active Low (AL) | 1 | 1 | 1 | 1 |
| 6 | EPWMxA Out = EPWMxA In (No Delay)<br>EPWMxB Out = EPWMxA In with Falling-Edge Delay | 0 or 1 | 0 or 1 | 0 | 1 |
| 7 | EPWMxA Out = EPWMxA In with Rising-Edge Delay<br>EPWMxB Out = EPWMxB In with No Delay | 0 or 1 | 0 or 1 | 1 | 0 |

**Table 15-9. Additional Dead-Band Operating Modes**

| Mode Description | DBCTL[DEDB-MODE] | DBCTL[OUTSWAP] | |
|---|---|---|---|
| | S8 | S6 | S7 |
| EPWMxA and EPWMxB signals are as defined by OUT-MODE bits. | 0 | 0 | 0 |
| EPWMxA = A-path as defined by OUT-MODE bits.<br>EPWMxB = A-path as defined by OUT-MODE bits (rising-edge delay or delay-bypassed A-signal path) | 0 | 0 | 1 |
| EPWMxA = B-path as defined by OUT-MODE bits (falling-edge delay or delay-bypassed B-signal path)<br>EPWMxB = B-path as defined by OUT-MODE bits | 0 | 1 | 0 |
| EPWMxA = B-path as defined by OUT-MODE bits (falling-edge delay or delay-bypassed B-signal path)<br>EPWMxB = A-path as defined by OUT-MODE bits (rising-edge delay or delay-bypassed A-signal path) | 0 | 1 | 1 |
| Rising-edge delay applied to EPWMxA / EPWMxB as selected by S4 switch (IN-MODE bits) on A signal path only.<br>Falling-edge delay applied to EPWMxA / EPWMxB as selected by S5 switch (IN-MODE bits) on B signal path only. | 0 | X | X |
| Rising-edge delay and falling-edge delay applied to source selected by S4 switch (IN-MODE bits) and output to B signal path only.(1) | 1 | X | X |

(1) When this bit is set to 1, 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.



Copyright © 2024 Texas Instruments Incorporated

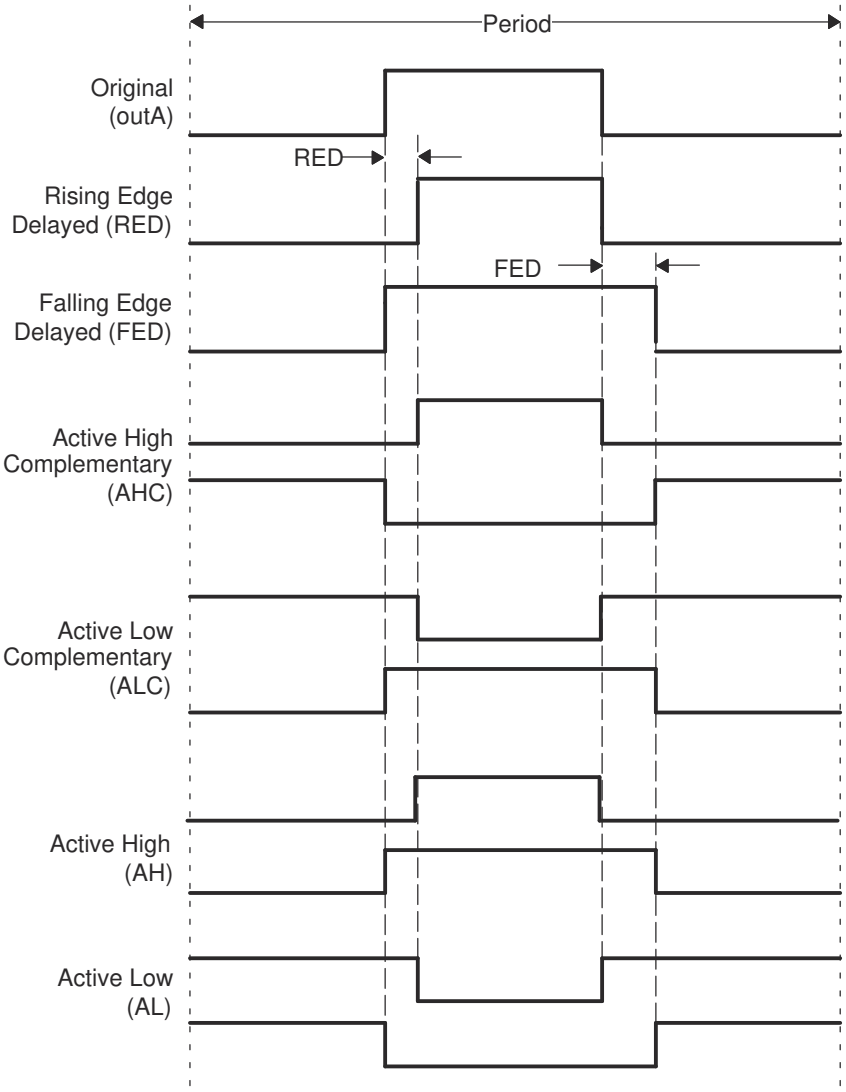



**Figure 15-35. Dead-Band Waveforms for Typical Cases (0% < Duty < 100%)**

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:

$FED = DBFED \times T_{TBCLK}$

$RED = DBRED \times T_{TBCLK}$

Where $T_{TBCLK}$ is the period of TBCLK, the prescaled version of EPWMCLK.



Copyright © 2024 Texas Instruments Incorporated

For convenience, delay values for various TBCLK options are shown in Table 15-10. The ePWM input clock frequency that these delay values been computed by is 100MHz.

**Table 15-10. Dead-Band Delay Values in µs as a Function of DBFED and DBRED**

| Dead-Band Value | Dead-Band Delay (µs) | | |
|---|---|---|---|
| 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:

$FED = DBFED \times T_{TBCLK}/2$

$RED = DBRED \times T_{TBCLK}/2$



Copyright © 2024 Texas Instruments Incorporated

## 15.8 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 15-36 illustrates the PWM chopper submodule within the ePWM.

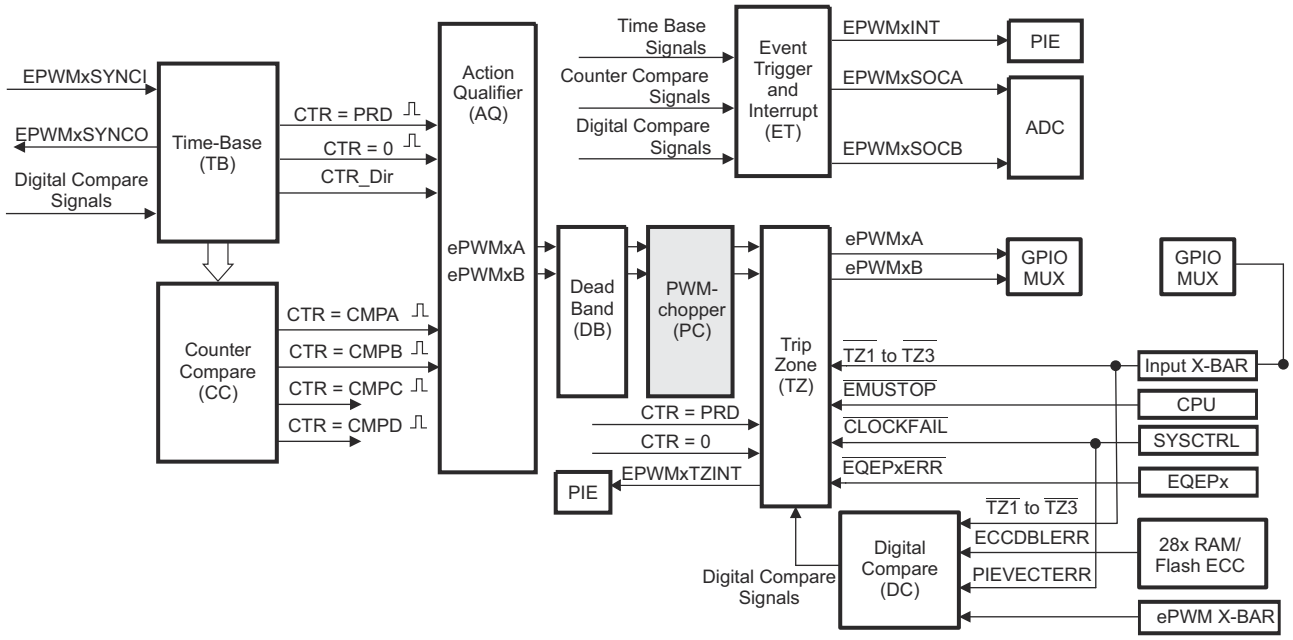



**Figure 15-36. PWM Chopper Submodule**

### *15.8.1 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

### *15.8.2 Operational Highlights for the PWM Chopper Submodule*

Figure 15-37 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.



Copyright © 2024 Texas Instruments Incorporated

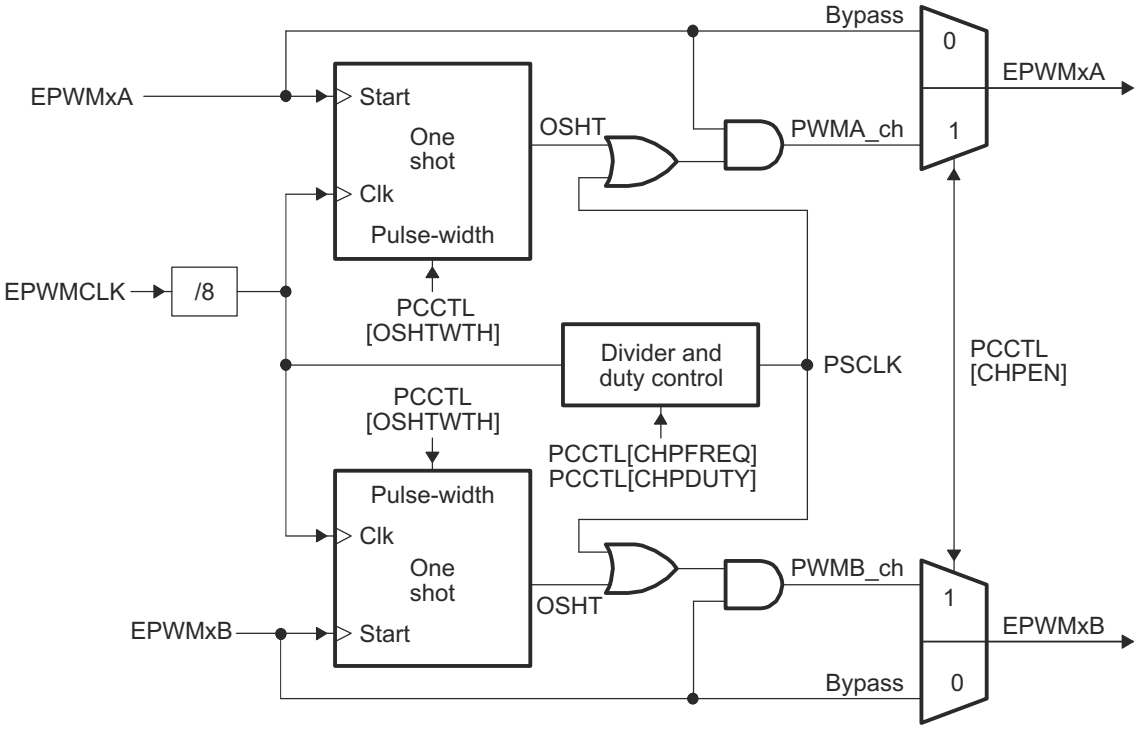



**Figure 15-37. PWM Chopper Submodule Operational Details**

### *15.8.3 Waveforms*

Figure 15-38 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.

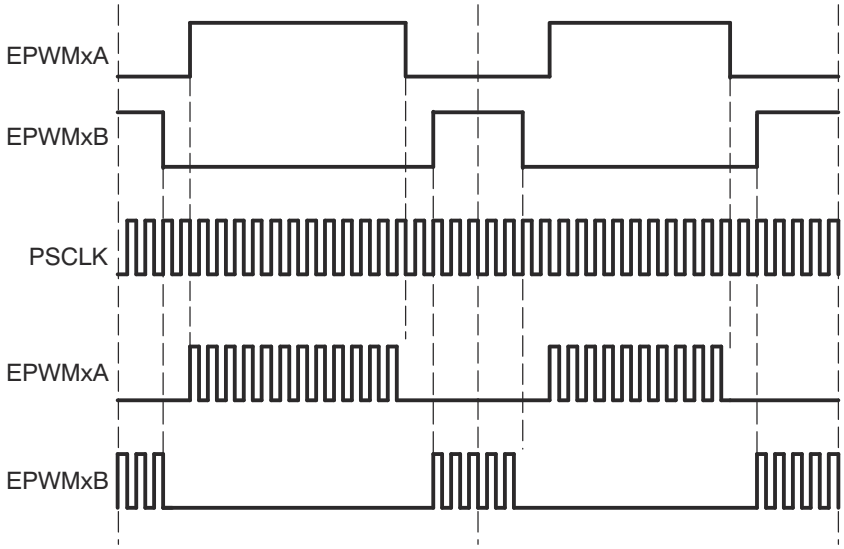



**Figure 15-38. Simple PWM Chopper Submodule Waveforms Showing Chopping Action Only**



Copyright © 2024 Texas Instruments Incorporated

#### 15.8.3.1 One-Shot Pulse

The width of the first pulse can be programmed to any of 16 possible pulse width values. The width or period of the first pulse is given by:

$$T_{1stpulse} = T_{EPWMCLK} \times 8 \times OSHTWTH$$

Where $T_{EPWMCLK}$ is the period of the system clock (EPWMCLK) and OSHTWTH is the four control bits (value from 1 to 16)

Figure 15-39 shows the first and subsequent sustaining pulses and Table 15-11 gives the possible pulse width values for a EPWMCLK = 80MHz.

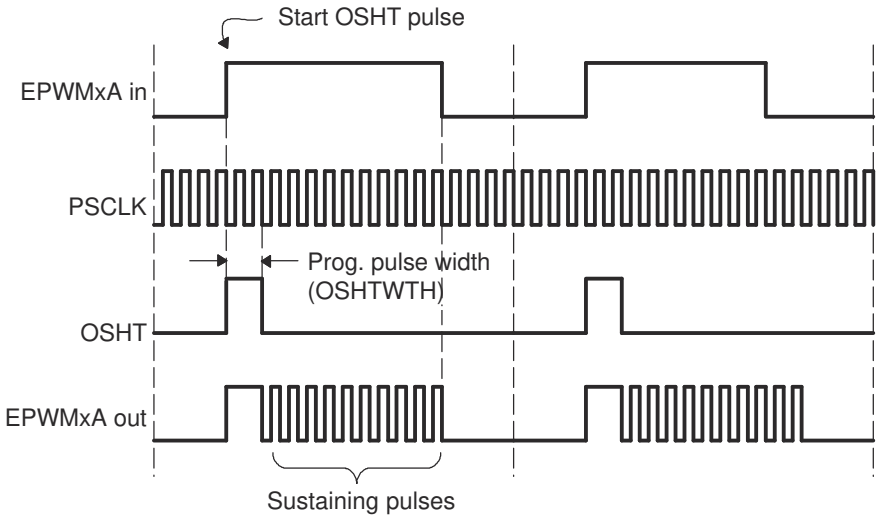



**Figure 15-39. PWM Chopper Submodule Waveforms Showing the First Pulse and Subsequent Sustaining Pulses**

**Table 15-11. 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 |



Copyright © 2024 Texas Instruments Incorporated

### 15.8.3.2 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 15-40 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%.

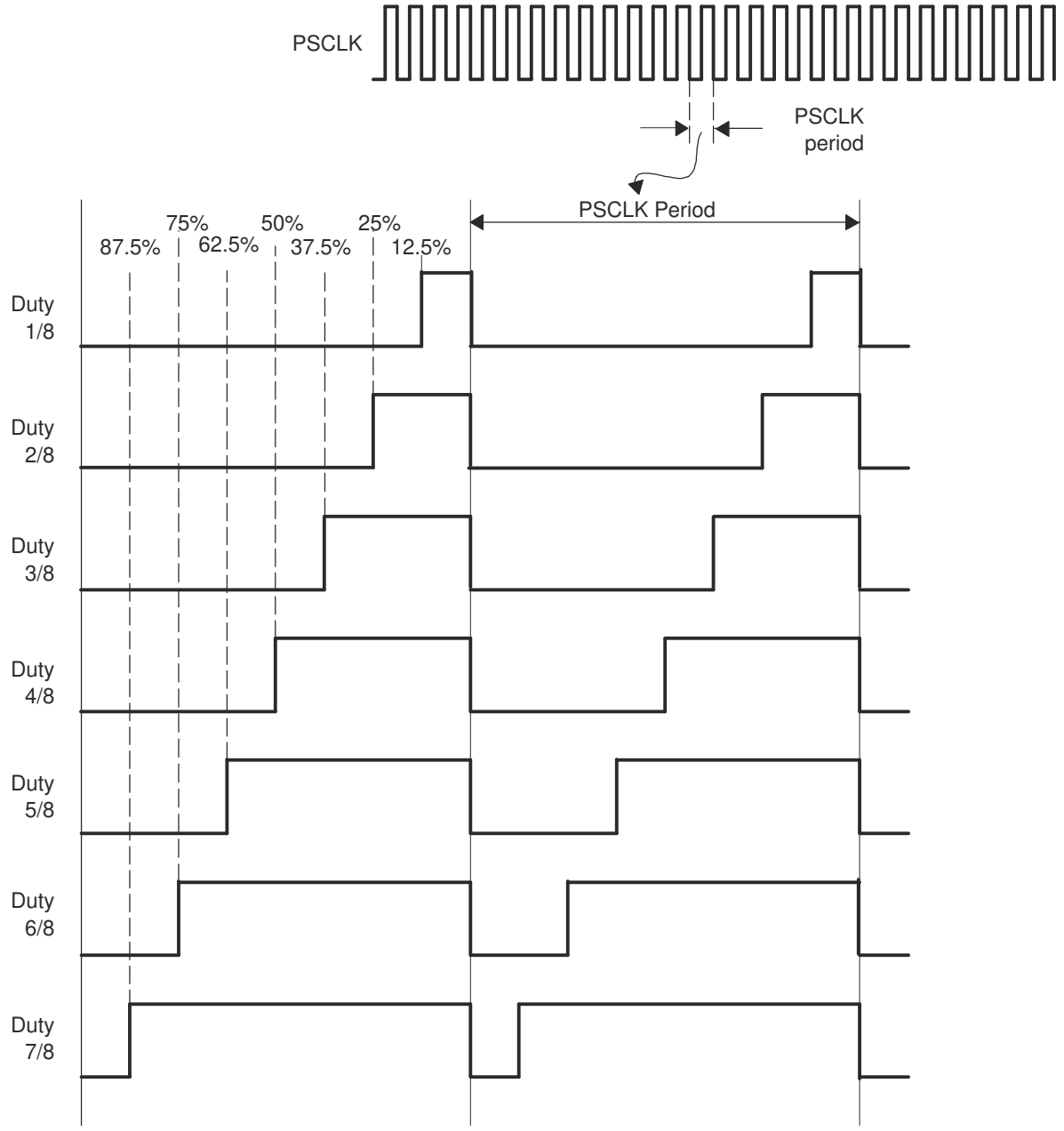



**Figure 15-40. PWM Chopper Submodule Waveforms Showing the Pulse Width (Duty Cycle) Control of Sustaining Pulses**



Copyright © 2024 Texas Instruments Incorporated

## 15.9 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 15-41 illustrates the trip-zone submodule within the ePWM.

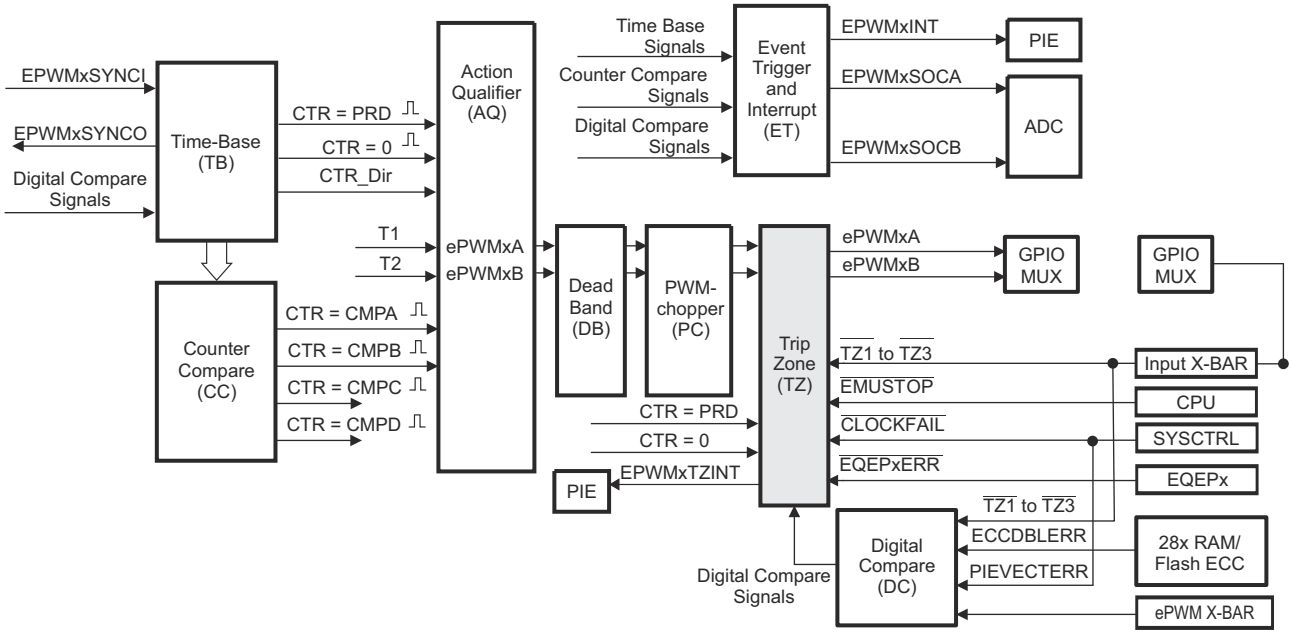



**Figure 15-41. Trip-Zone Submodule**

### *15.9.1 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.



Copyright © 2024 Texas Instruments Incorporated

### 15.9.2 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*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 15-12 lists some of the possible actions. Independent actions can be specified based on the occurrence of the event while the counter is counting up 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 15-12 lists some of the possible actions. Independent actions can be specified based on the occurrence of the event while the counter is counting up and 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.



Copyright © 2024 Texas Instruments Incorporated

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 15.11.

  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 15-12 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 15-12, can be taken on a trip event.

**Table 15-12. 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.<br>No change is made to the output. |



Copyright © 2024 Texas Instruments Incorporated

***Example 15-1. 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.

### *15.9.3 Generating Trip Event Interrupts*

Figure 15-42 and Figure 15-43 illustrate the trip-zone submodule control and interrupt logic, respectively. DCAEVT1/2 and DCBEVT1/2 signals are described in further detail in Section 15.11.



Copyright © 2024 Texas Instruments Incorporated

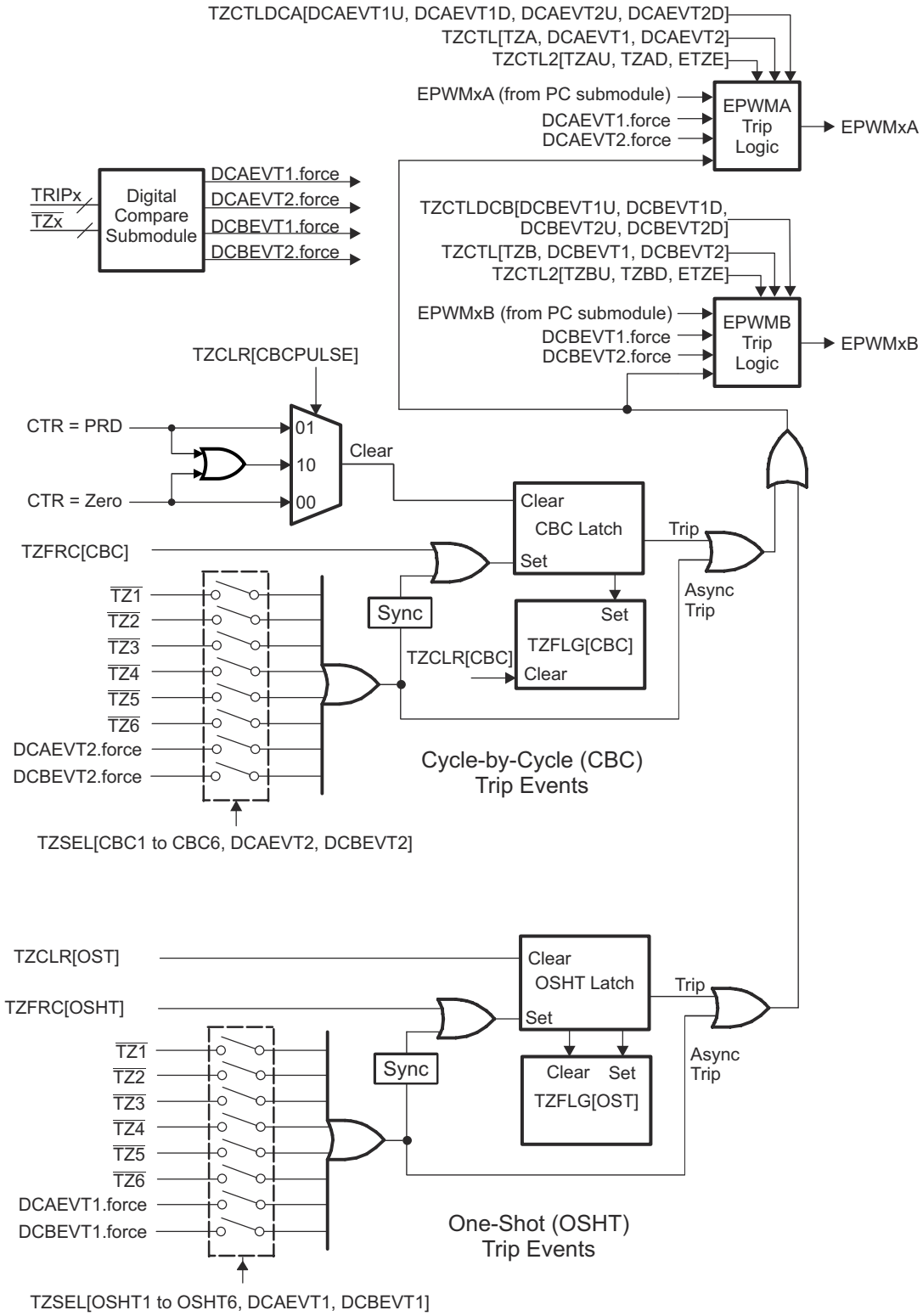



**Figure 15-42. Trip-Zone Submodule Mode Control Logic**



Copyright © 2024 Texas Instruments Incorporated

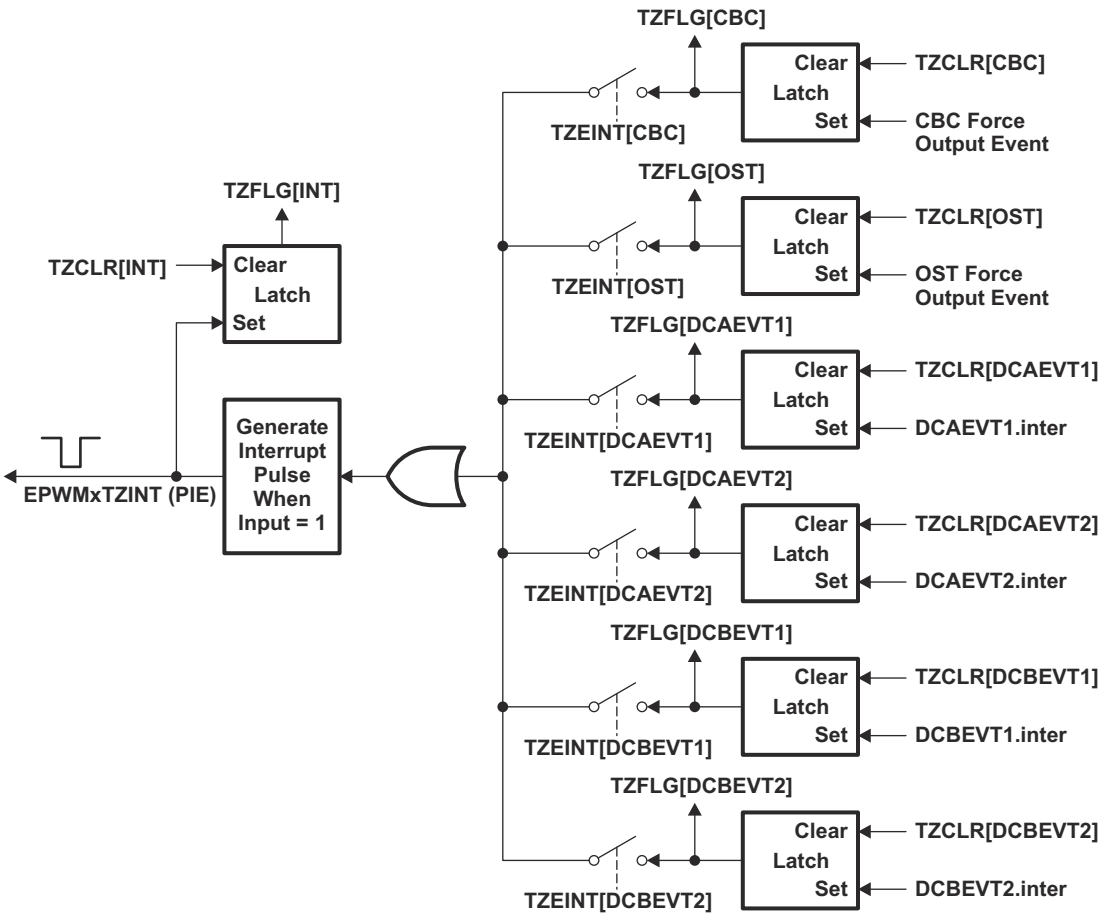



**Figure 15-43. Trip-Zone Submodule Interrupt Logic**

These individual flags for the CBC, OST and DCxEVTy can be used to detect the source of the EPWMxTZINT Interrupt. When multiple sources are used to generate the EPWMxTZINT interrupt, reading and clearing the flags takes different actions based on the specific event.



Copyright © 2024 Texas Instruments Incorporated

## 15.10 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 15-44 illustrates the event-trigger submodule within the ePWM.

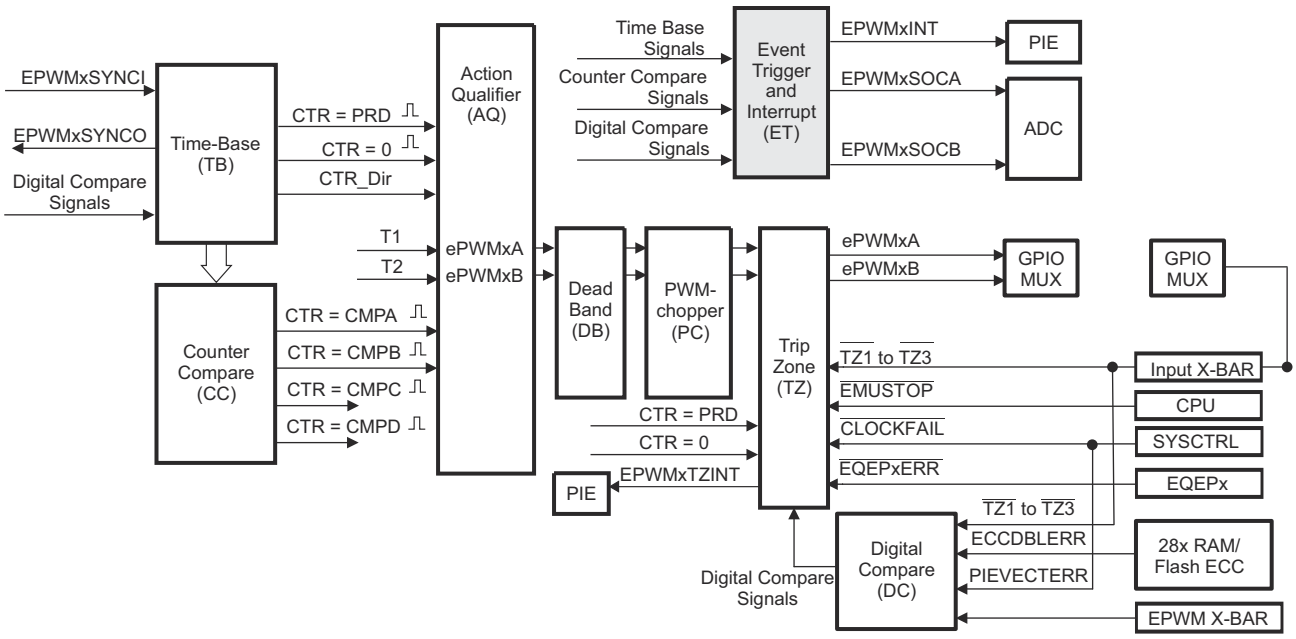



**Figure 15-44. Event-Trigger Submodule**



Copyright © 2024 Texas Instruments Incorporated

### 15.10.1 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 15-45) 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

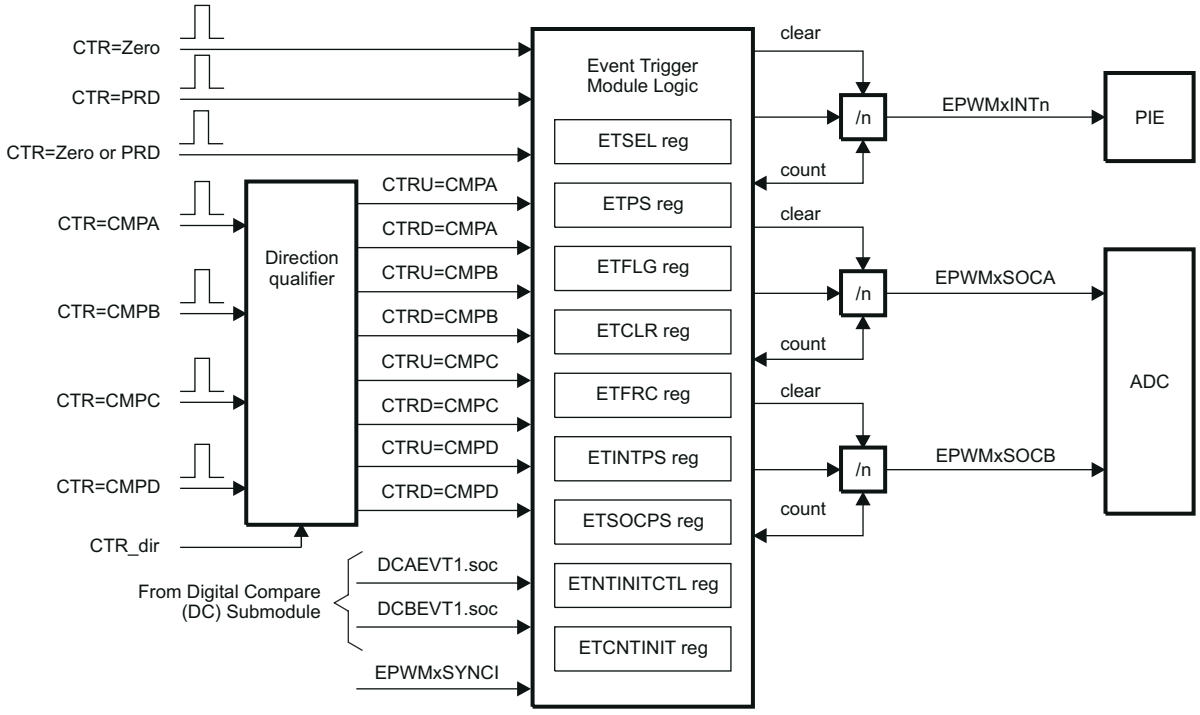



**Figure 15-45. Event-Trigger Submodule Showing Event Inputs and Prescaled Outputs**

- 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 15-46, Figure 15-47, and Figure 15-48.

Figure 15-46 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.



Copyright © 2024 Texas Instruments Incorporated

The selection made on ETPS[INTPSSEL] bit determines whether ETINTPS register, INTCNT2 and INTPRD2 bit fields determine frequency of events (interrupt once every 0-15 events).

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[INTPSSEL]. 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[INTPSSEL]. 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].



Copyright © 2024 Texas Instruments Incorporated

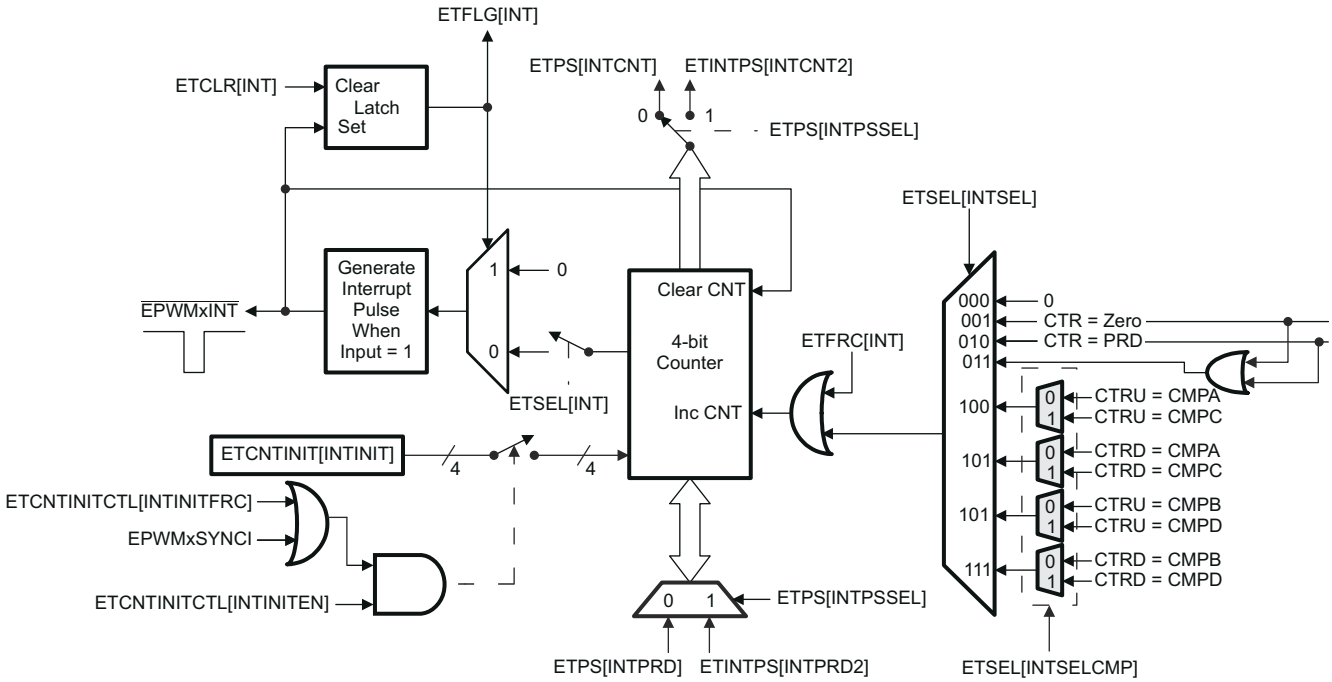
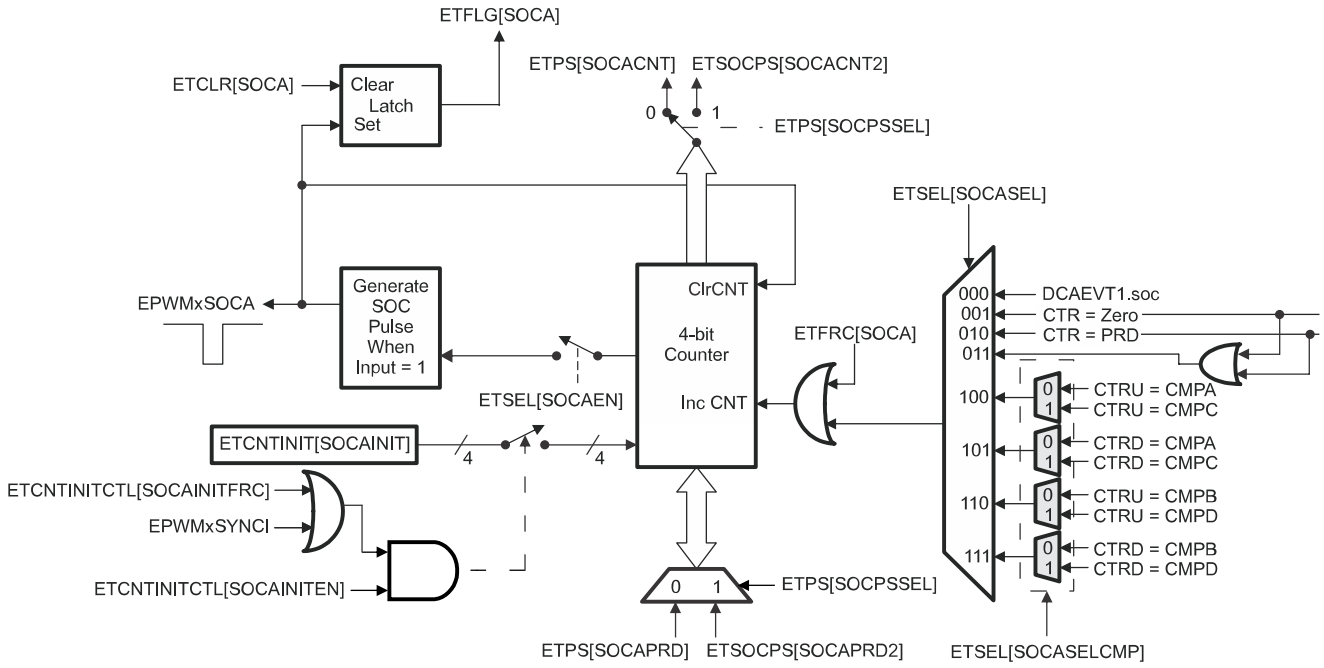



**Figure 15-46. Event-Trigger Interrupt Generator**

Figure 15-47 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[SOCAEN] stops pulse generation, but input events can still be counted until the period value is reached as with the interrupt generation logic. The event that 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.



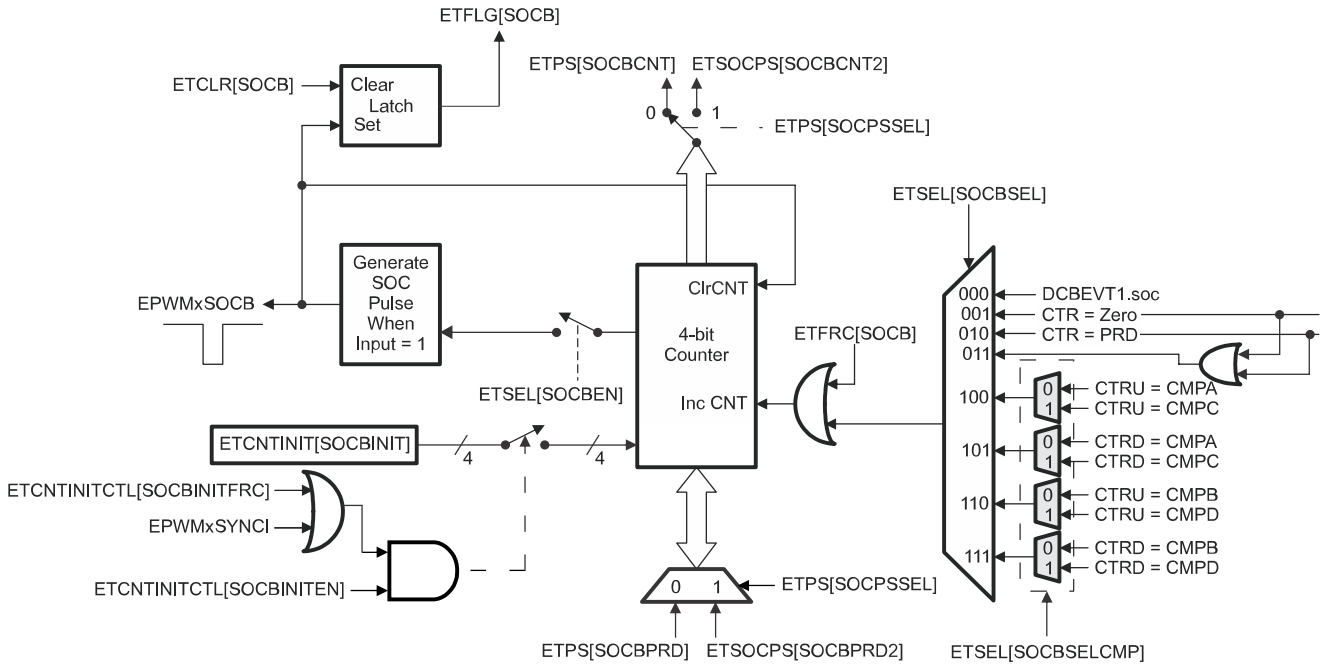
Copyright © 2024 Texas Instruments Incorporated




NOTE: The DCAEVT1.soc signals are generated by the Digital Compare (DC) submodule in Section 15.11.

**Figure 15-47. Event-Trigger SOCA Pulse Generator**

Figure 15-48 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 15.11.

**Figure 15-48. Event-Trigger SOCB Pulse Generator**



Copyright © 2024 Texas Instruments Incorporated

## 15.11 Digital Compare (DC) Submodule

Figure 15-49 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 15-50.

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.

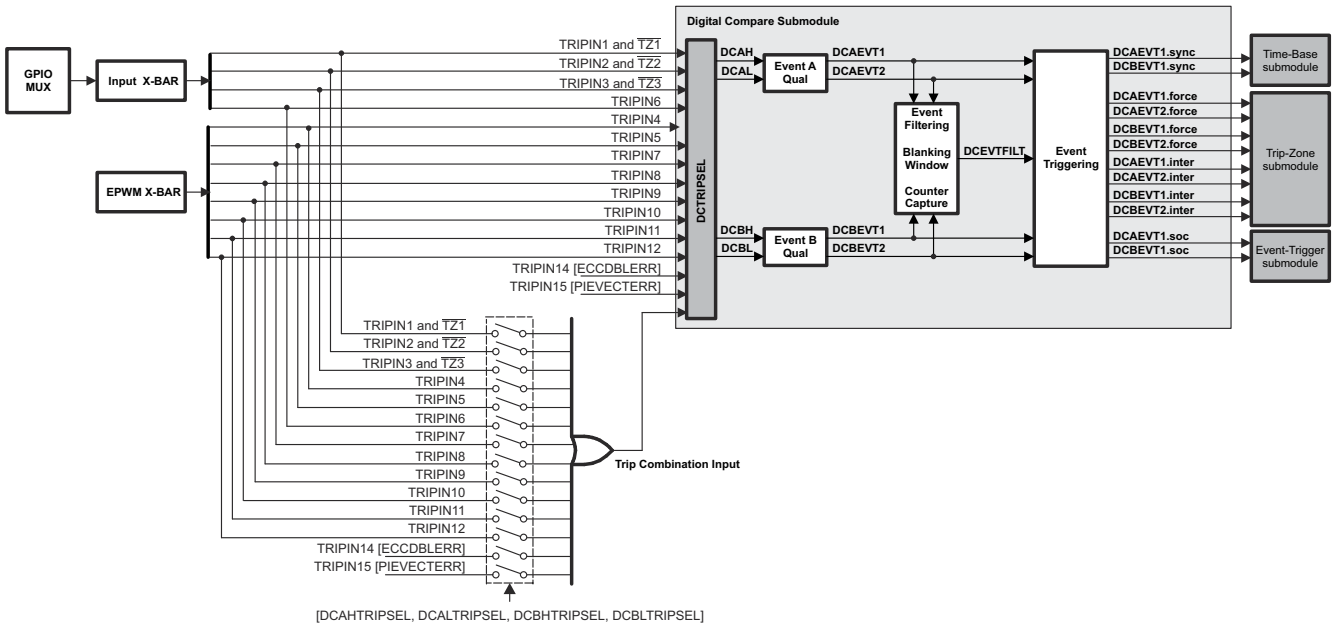



**Figure 15-49. Digital-Compare Submodule High-Level Block Diagram**



Copyright © 2024 Texas Instruments Incorporated

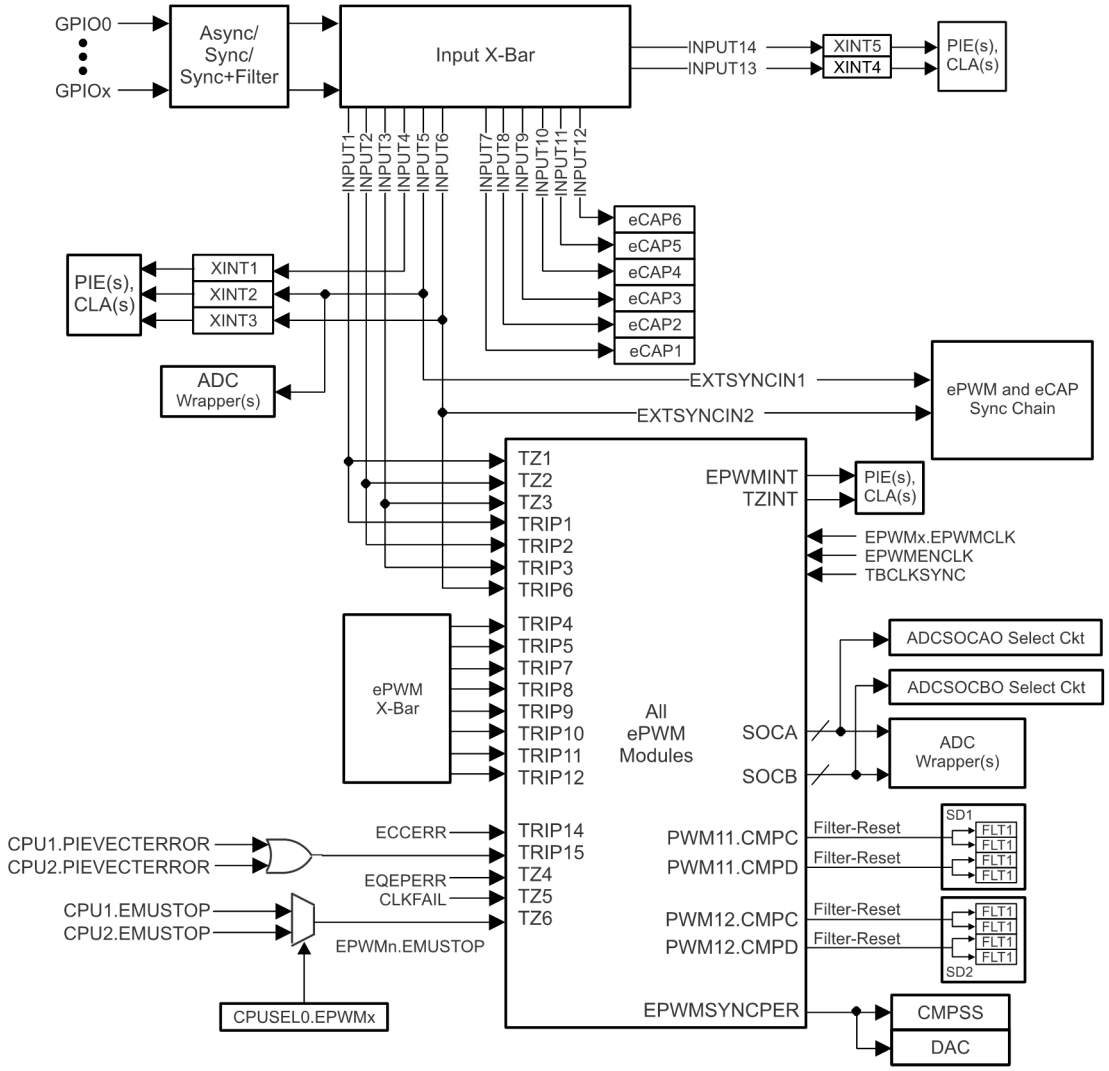



**Figure 15-50. GPIO MUX-to-Trip Input Connectivity**



Copyright © 2024 Texas Instruments Incorporated

### 15.11.1 Purpose of the Digital Compare Submodule

The key functions of the digital compare submodule are:

- Analog comparator (COMP) module outputs fed though 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.

### 15.11.2 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.

### 15.11.3 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.



Copyright © 2024 Texas Instruments Incorporated

### 15.11.4 Operation Highlights of the Digital Compare Submodule

The following sections describe the operational highlights and configuration options for the digital compare submodule.

#### 15.11.4.1 Digital Compare Events

As described in Section 15.11.1, 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 * TBCLK sync pulse width is required. If pulse width is < 3*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 EPWMxSYNCI input signal and the TBCTL[SWFSYNC] signal to generate a synchronization pulse to the time-base counter.



Copyright © 2024 Texas Instruments Incorporated

Figure 15-51 and Figure 15-52 show how the DCAEVT1, DCAEVT2, or DCEVTFLT signals are processed to generate the digital compare A event force, interrupt, soc, and sync signals.

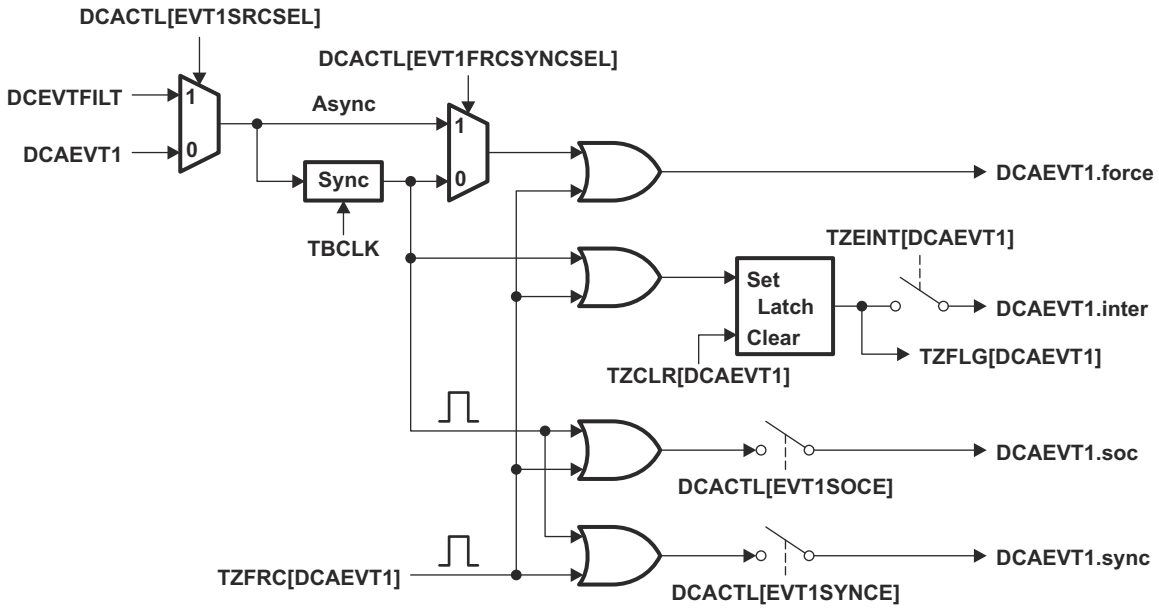



**Figure 15-51. DCAEVT1 Event Triggering**

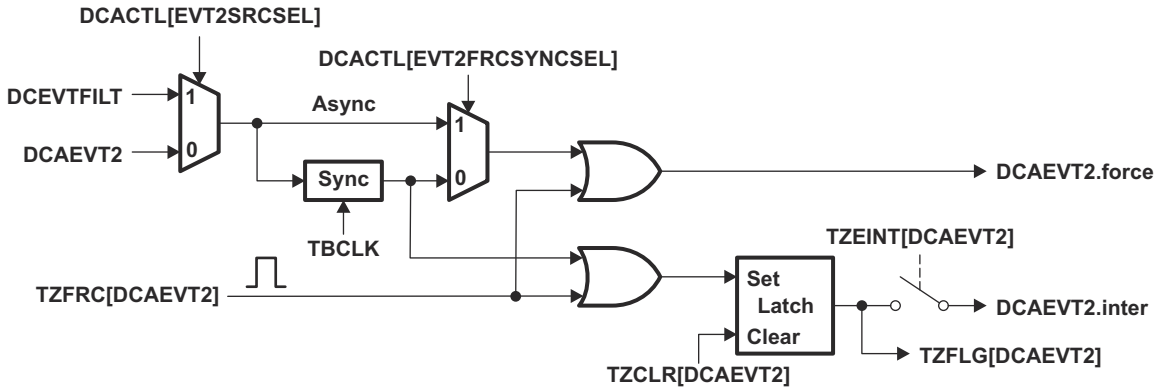



**Figure 15-52. DCAEVT2 Event Triggering**



Copyright © 2024 Texas Instruments Incorporated

Figure 15-53 and Figure 15-54 show how the DCBEVT1, DCBEVT2, or DCEVTFLT signals are processed to generate the digital compare B event force, interrupt, soc, and sync signals.

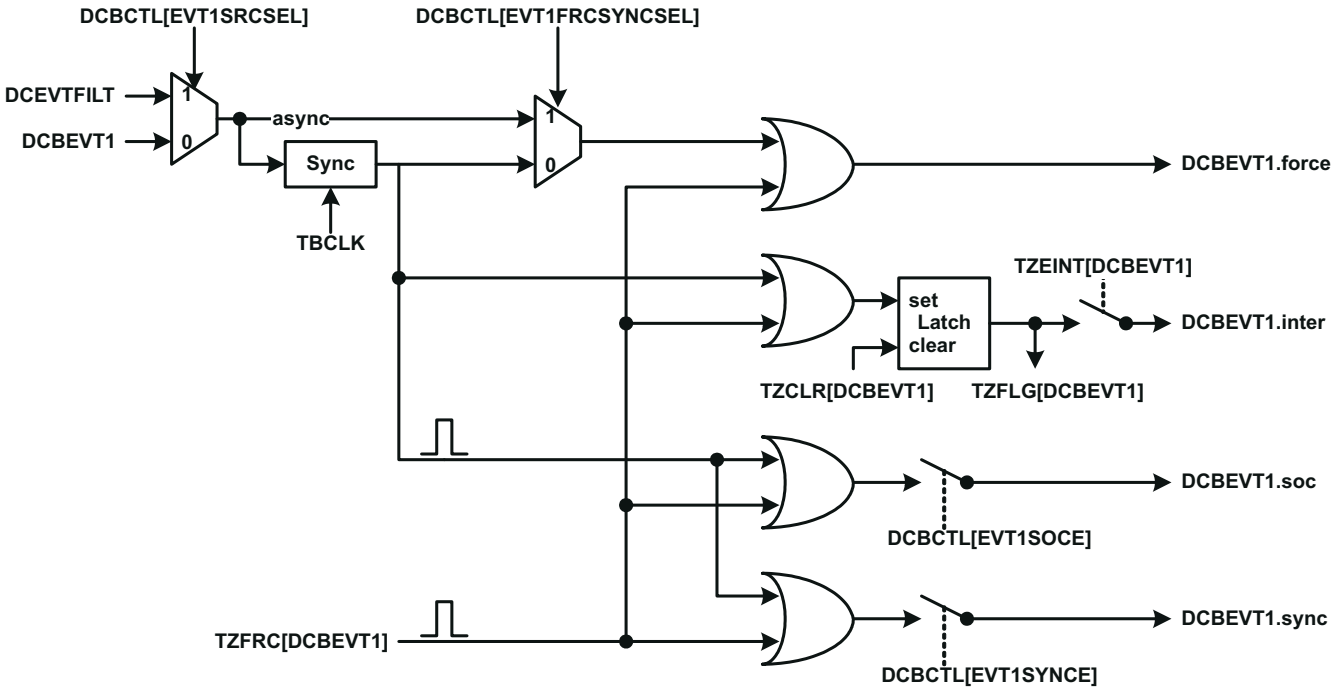



**Figure 15-53. DCBEVT1 Event Triggering**

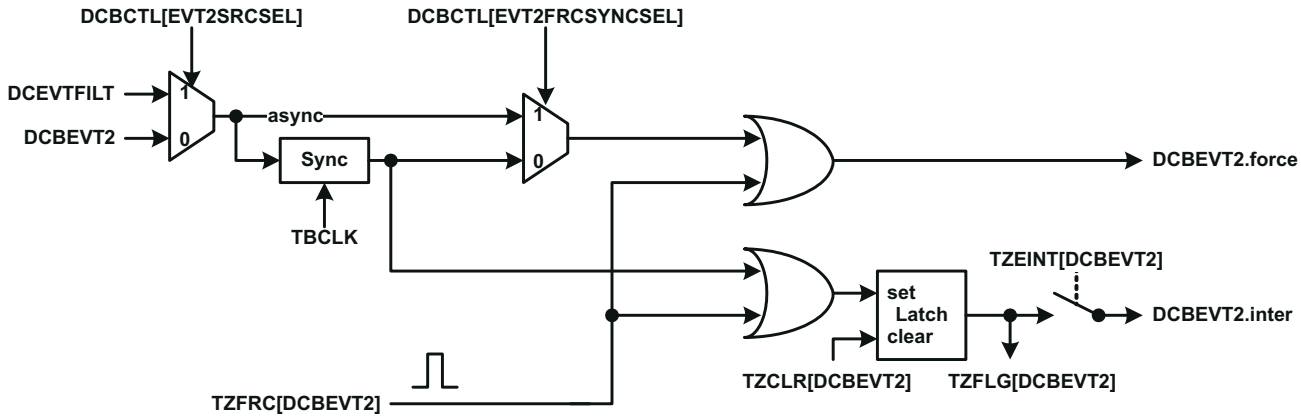


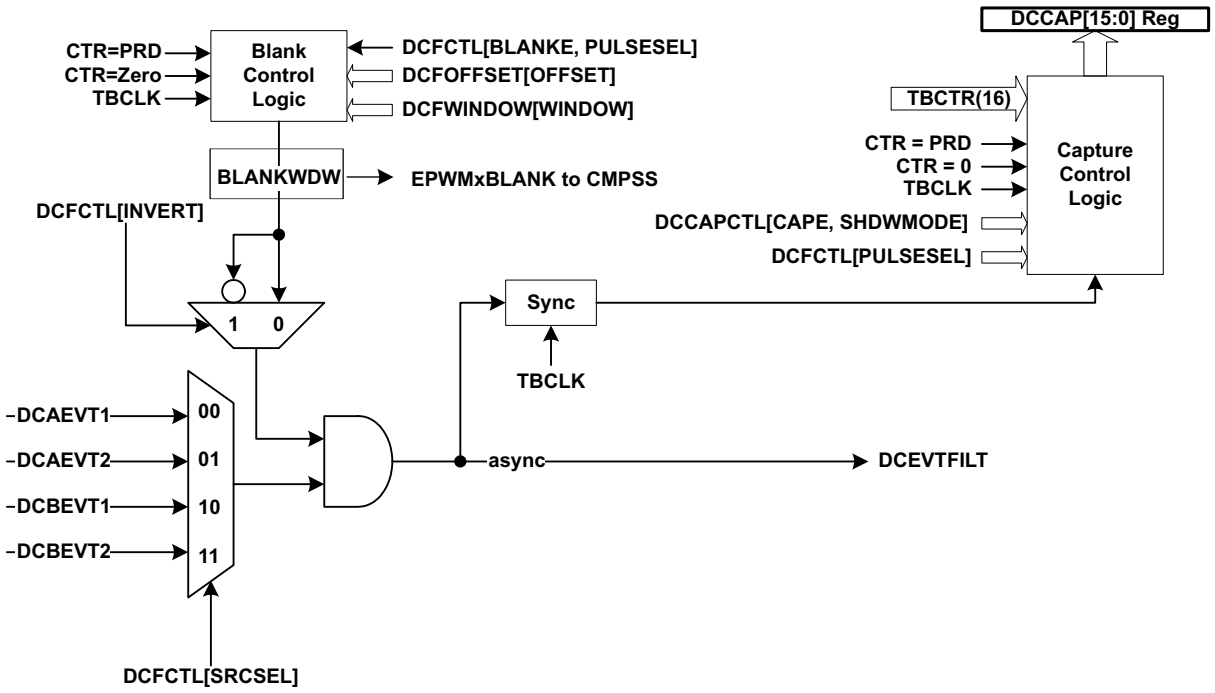

**Figure 15-54. DCBEVT2 Event Triggering**



Copyright © 2024 Texas Instruments Incorporated

### 15.11.4.2 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 DCxEVTy 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 by the application. Before and after the blanking window ends, events can generate soc, sync, interrupt, and force signals as before. Figure 15-55 shows the details of the event filtering logic.




A. On the F2837x/F2807x family of devices, EPWMxBLANK does not go to the CMPSS

**Figure 15-55. Event Filtering**

**Capture Control Logic:** The event filtering can also capture the TBCTR value of the selected DCxEVTy event as configured in the DCCAPCTL register. When capture control logic is enabled, the selected DCxEVTy 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 DCFCCTL[PULSESEL], and the CPU reads from the shadow register. After the selected DCxEVTy 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].



Copyright © 2024 Texas Instruments Incorporated

Figure 15-56 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.

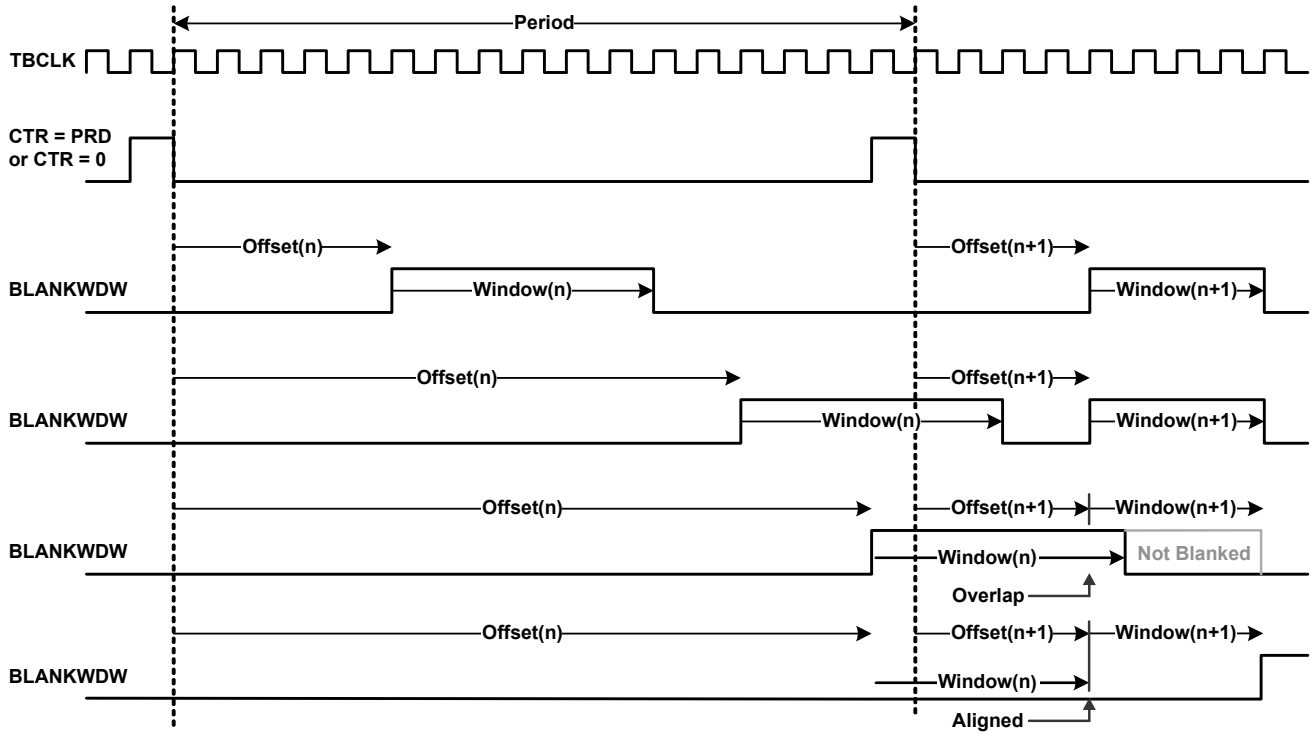



**Figure 15-56. Blanking Window Timing Diagram**



Copyright © 2024 Texas Instruments Incorporated

#### 15.11.4.3 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 DCxEVTy 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 DCxEVTy 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 DCxEVTy 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 15-57.

**Note**

A specific application example showcasing the usage of valley switching hardware and software is available in C2000Ware.



Copyright © 2024 Texas Instruments Incorporated

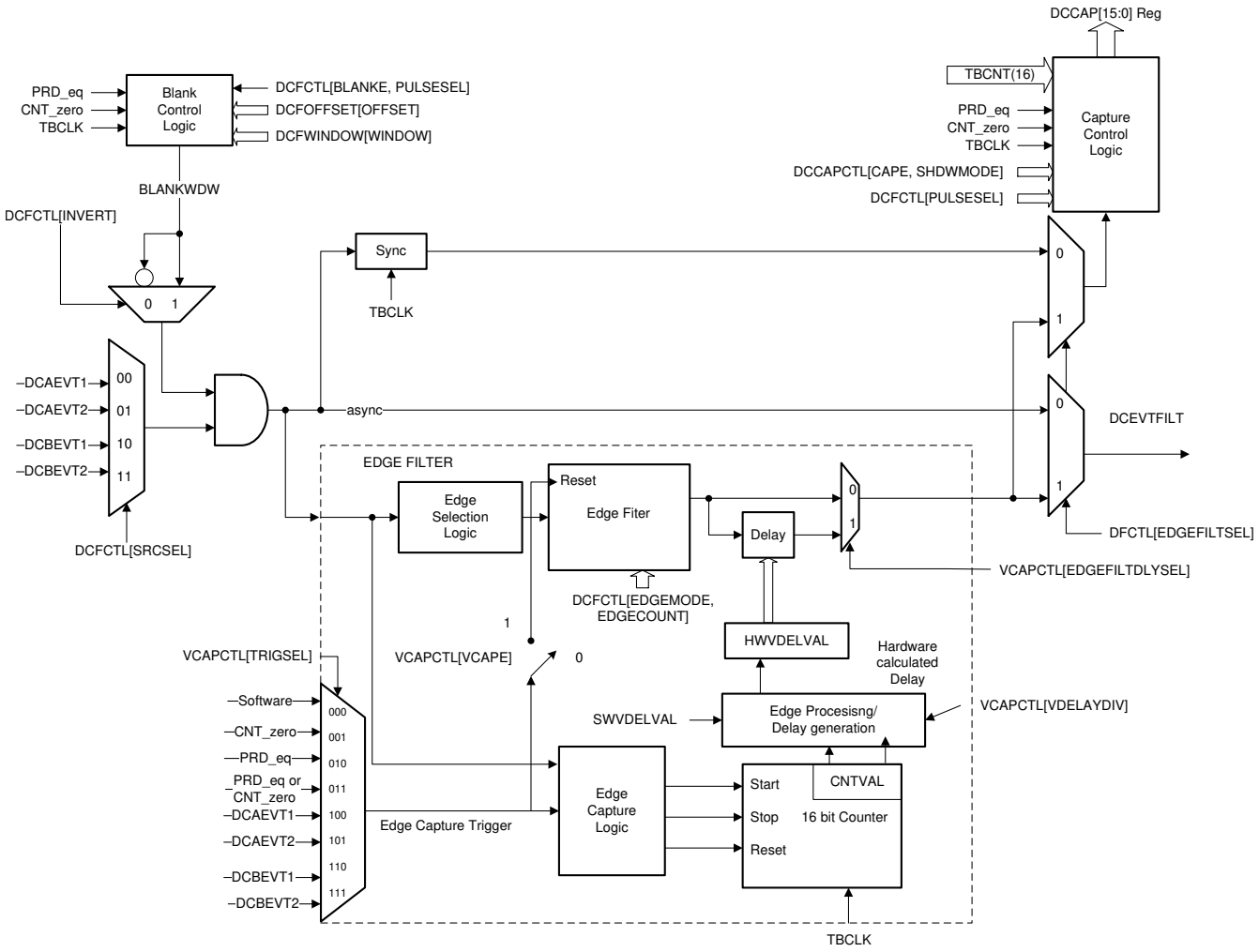



**Figure 15-57. Valley Switching**



Copyright © 2024 Texas Instruments Incorporated

## 15.12 ePWM Crossbar (X-BAR)

Figure 15-58 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.

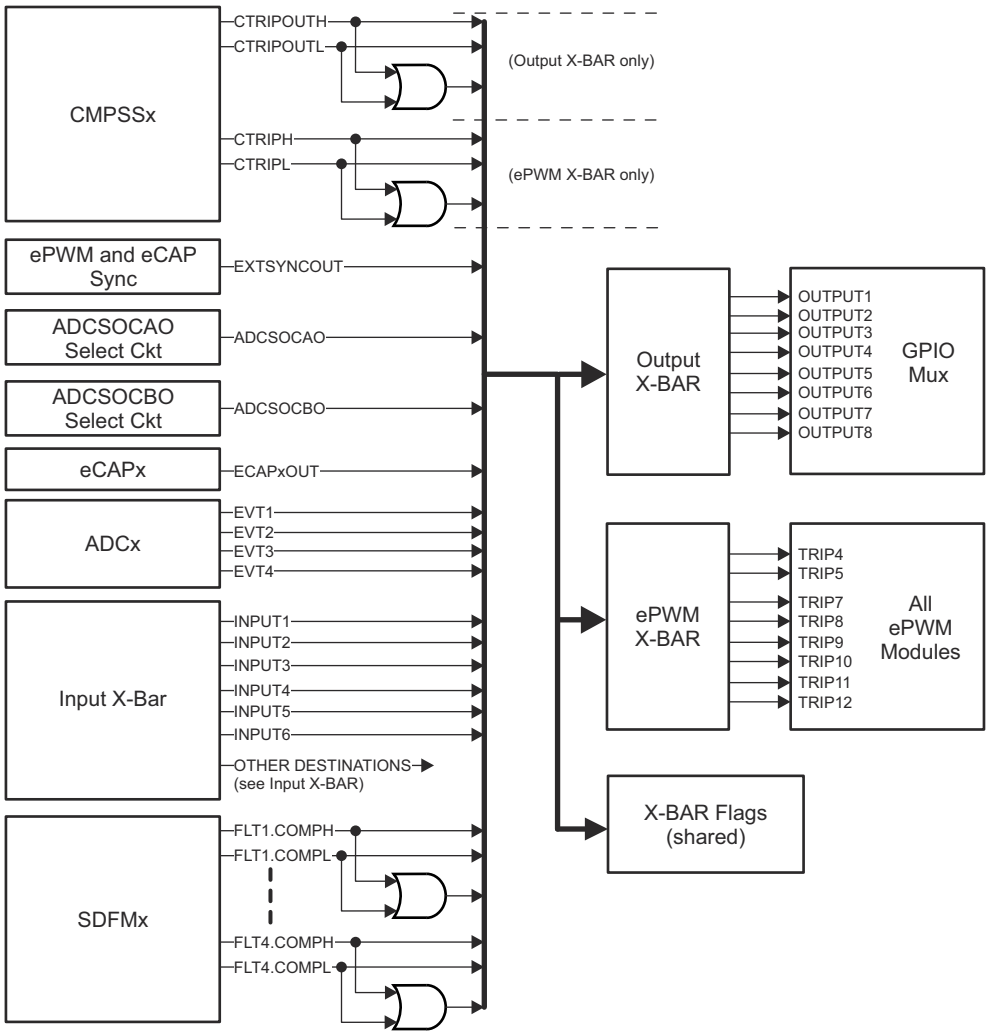



**Figure 15-58. ePWM X-BAR**



Copyright © 2024 Texas Instruments Incorporated

## 15.13 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.

### *15.13.1 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 15-59. This simplified ePWM block shows only the key resources needed to explain how a multiswitch power topology is controlled with multiple ePWM modules working together.

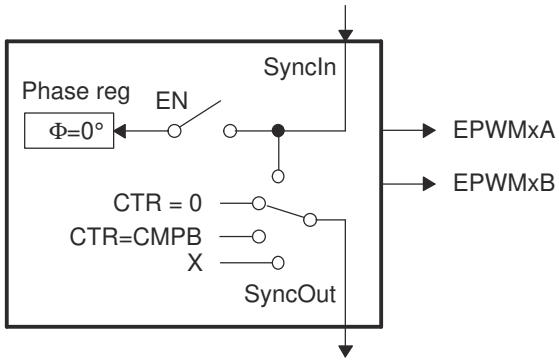



**Figure 15-59. Simplified ePWM Module**



Copyright © 2024 Texas Instruments Incorporated

### 15.13.2 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 15-60.

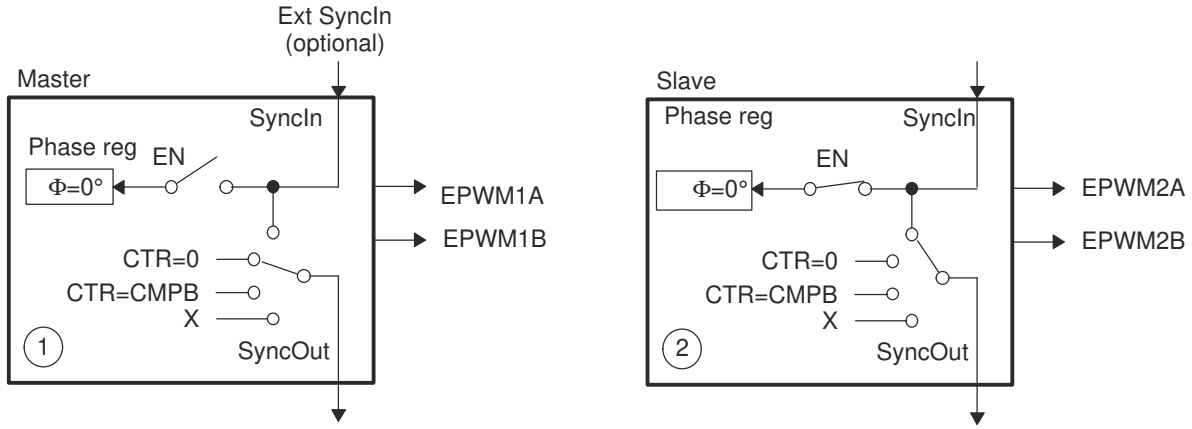


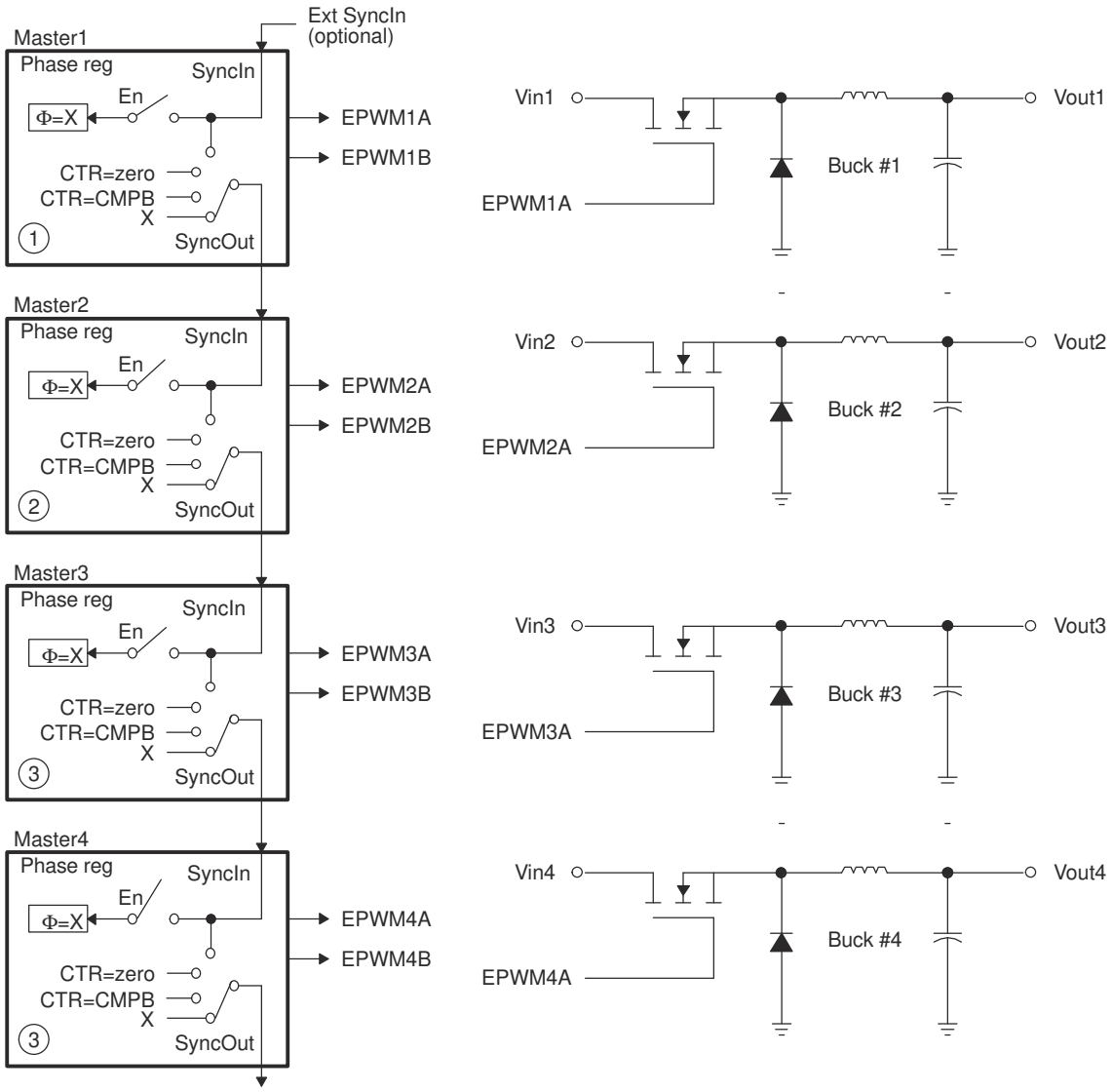

**Figure 15-60. EPWM1 Configured as a Typical Master, EPWM2 Configured as a Slave**



Copyright © 2024 Texas Instruments Incorporated

### 15.13.3 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 15-61 shows four buck stages, each running at independent frequencies. In this case, all four ePWM modules are configured as Masters and no synchronization is used. Figure 15-62 shows the waveforms generated by the setup shown in Figure 15-61; note that only three waveforms are shown, although there are four stages.




A. φ = X indicates value in phase register is a "don't care"

**Figure 15-61. Control of Four Buck Stages. Here $F_{PWM1} \neq F_{PWM2} \neq F_{PWM3} \neq F_{PWM4}$**



Copyright © 2024 Texas Instruments Incorporated

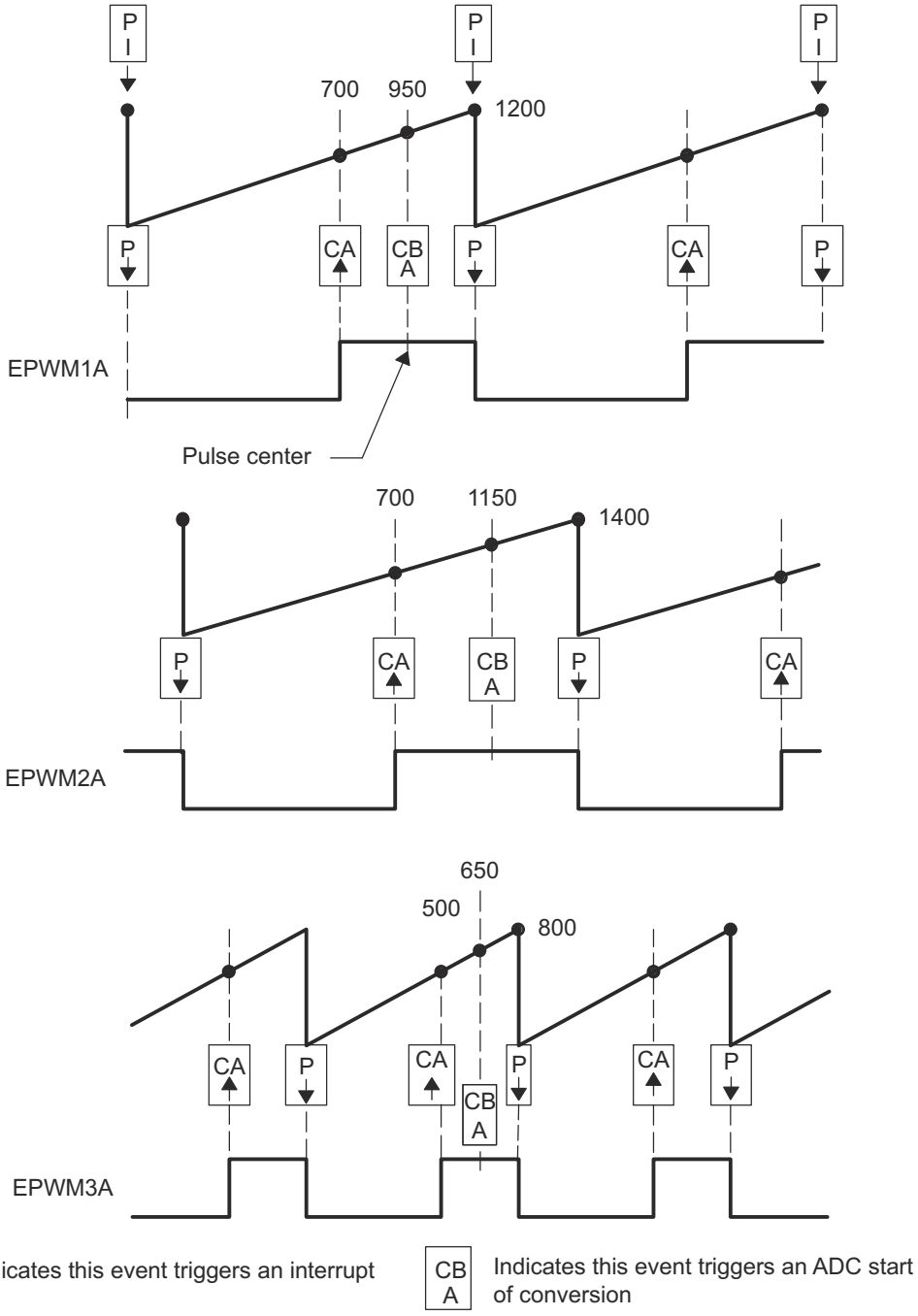
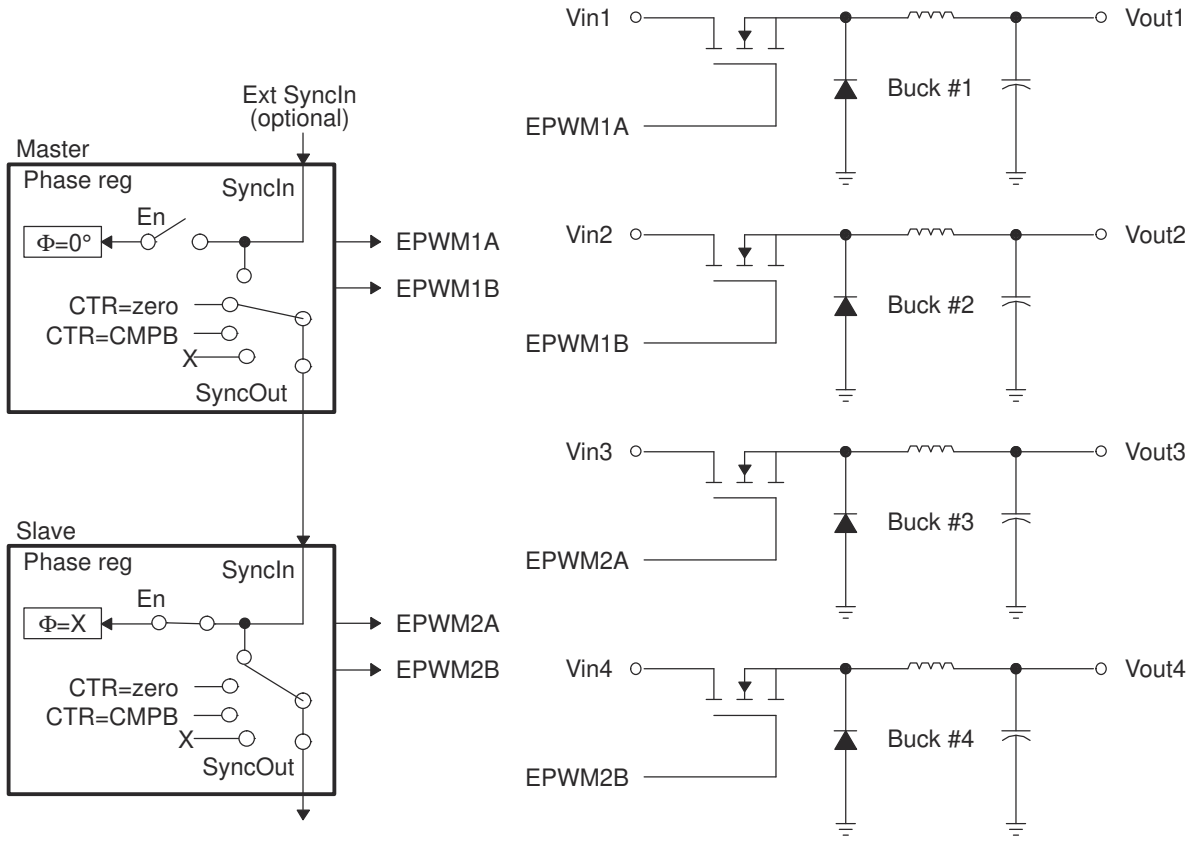



**Figure 15-62. Buck Waveforms for Control of Four Buck Stages (Note: Only three bucks shown here)**


Copyright © 2024 Texas Instruments Incorporated

### 15.13.4 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 15-63 shows such a configuration; Figure 15-64 shows the waveforms generated by the configuration.

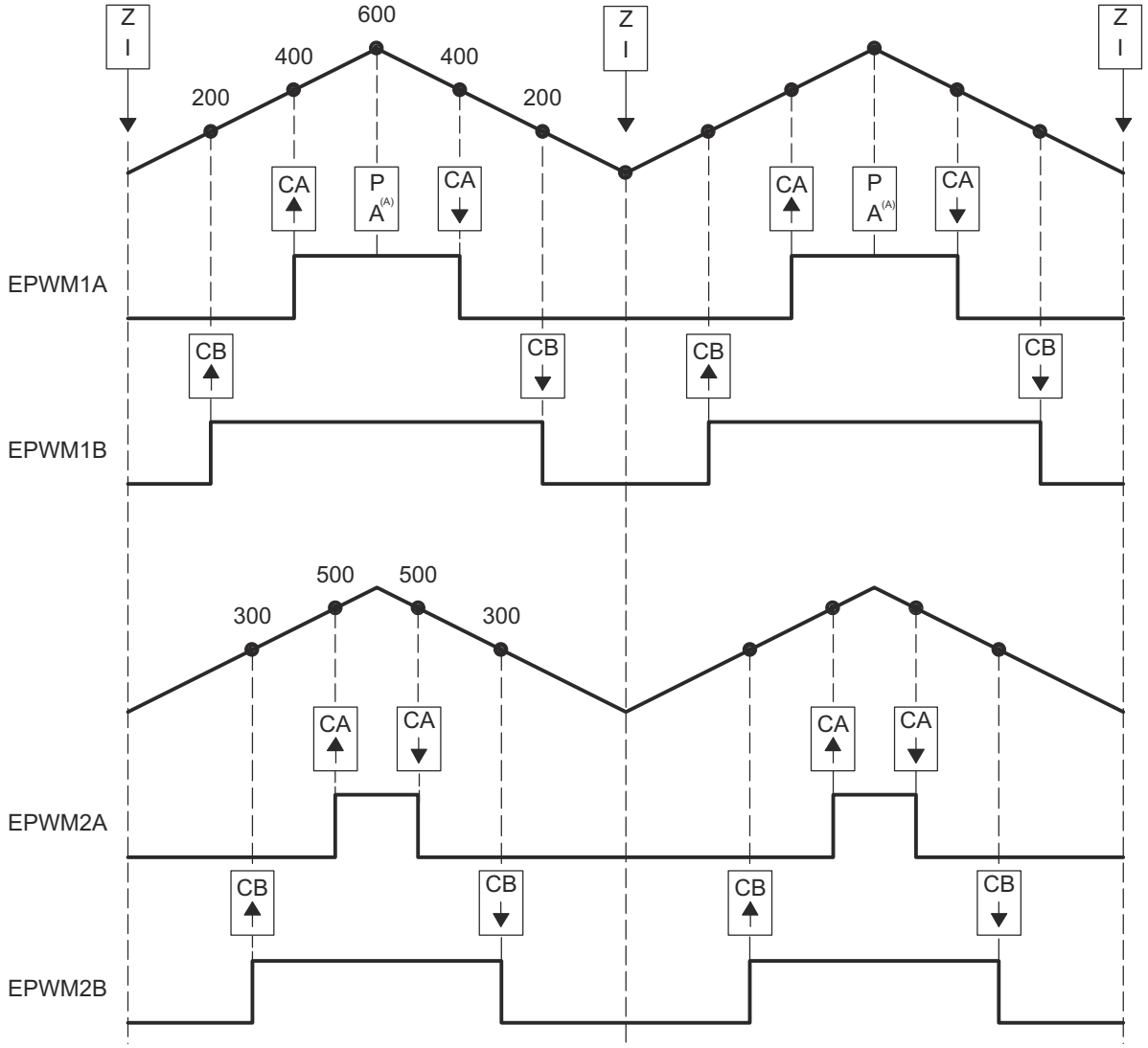



A. φ = X indicates value in phase register is a "don't care"

**Figure 15-63. Control of Four Buck Stages. (Note: $F_{PWM2}$ = N x $F_{PWM1}$)**



Copyright © 2024 Texas Instruments Incorporated




A. Starts ADC conversion.

**Figure 15-64. Buck Waveforms for Control of Four Buck Stages (Note: $F_{PWM2} = F_{PWM1}$)**



Copyright © 2024 Texas Instruments Incorporated

### 15.13.5 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 15-65 shows control of two synchronized Half-H bridge stages where stage 2 can operate at integer multiple (N) frequencies of stage 1. Figure 15-66 shows the waveforms generated by the configuration shown in Figure 15-65.

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.

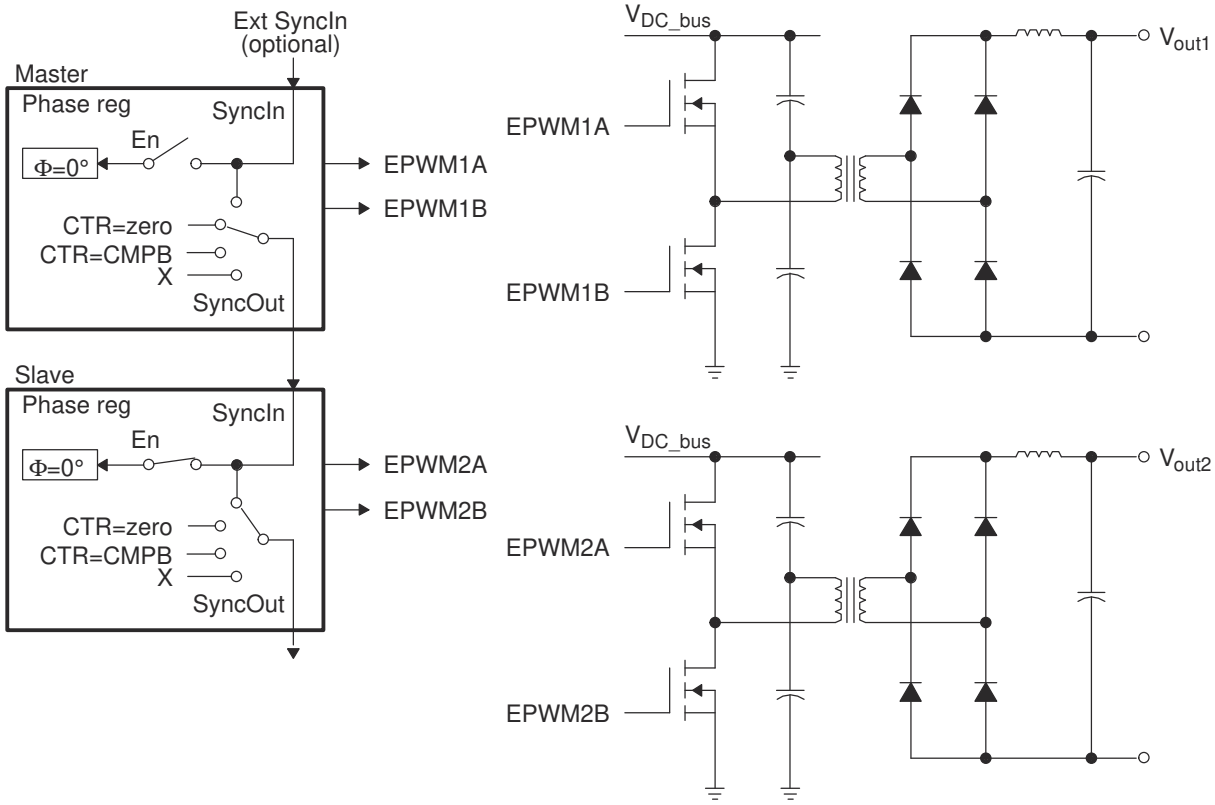



**Figure 15-65. Control of Two Half-H Bridge Stages ($F_{PWM2}$ = N x $F_{PWM1}$)**



Copyright © 2024 Texas Instruments Incorporated

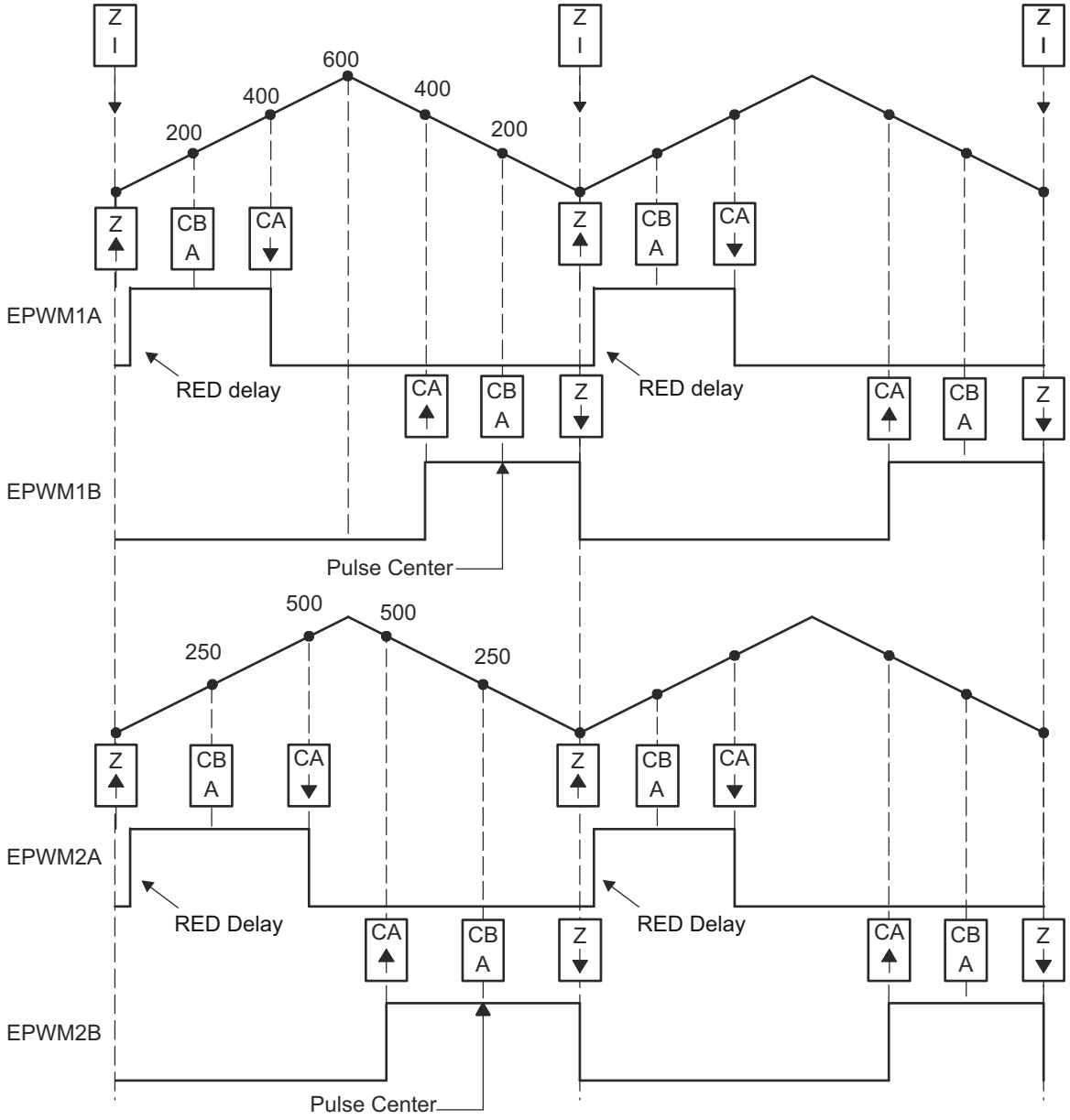



**Figure 15-66. Half-H Bridge Waveforms for Control of Two Half-H Bridge Stages (Note: Here $F_{PWM2} = F_{PWM1}$)**


Copyright © 2024 Texas Instruments Incorporated

### 15.13.6 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 15-67 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 15-67), or both inverters can be synchronized by using one master (module 1) and five slaves. In this case, the frequency of modules 4, 5, and 6 (all equal) can be integer multiples of the frequency for modules 1, 2, and 3 (also all equal).

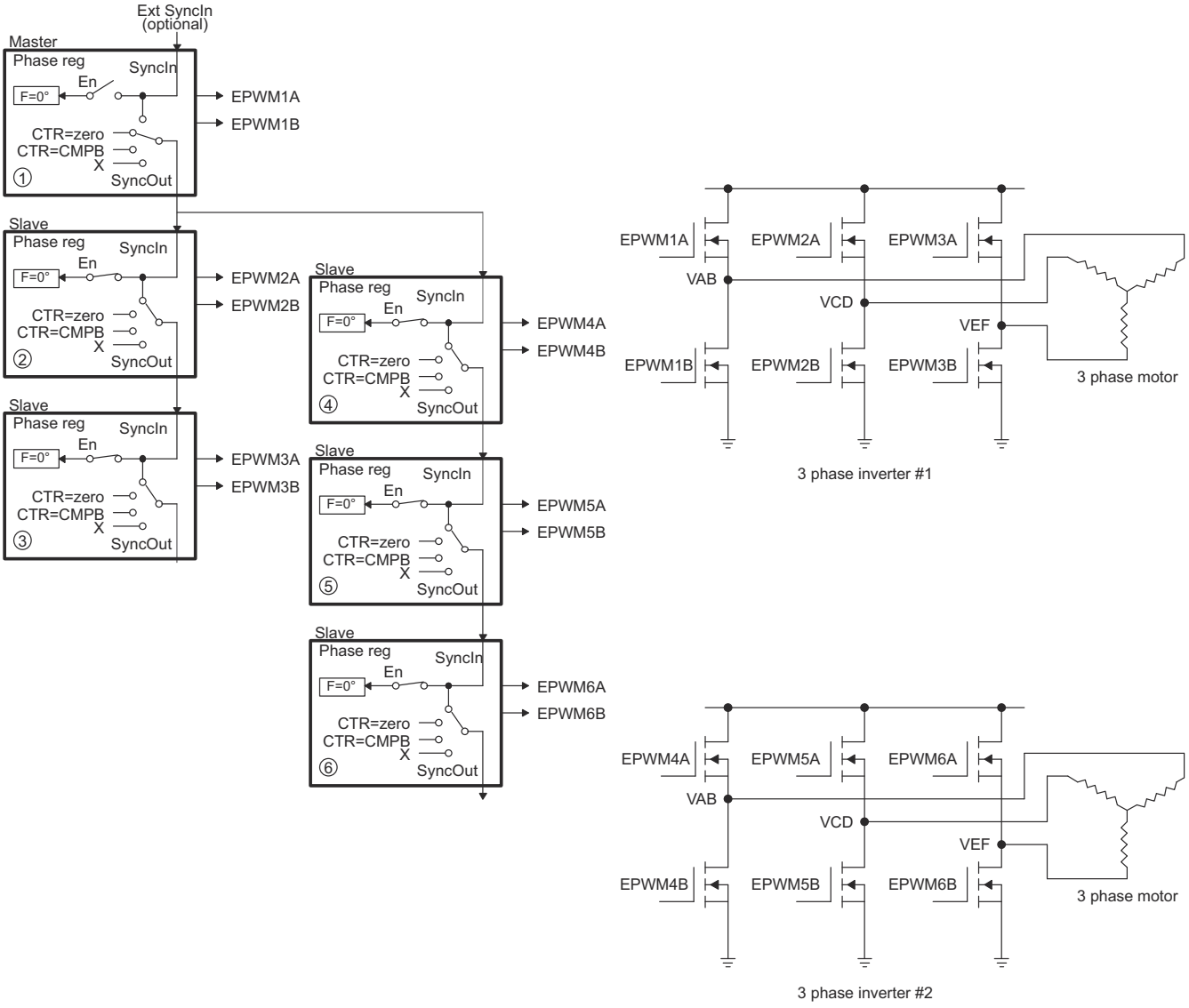



**Figure 15-67. Control of Dual 3-Phase Inverter Stages as Is Commonly Used in Motor Control**



Copyright © 2024 Texas Instruments Incorporated

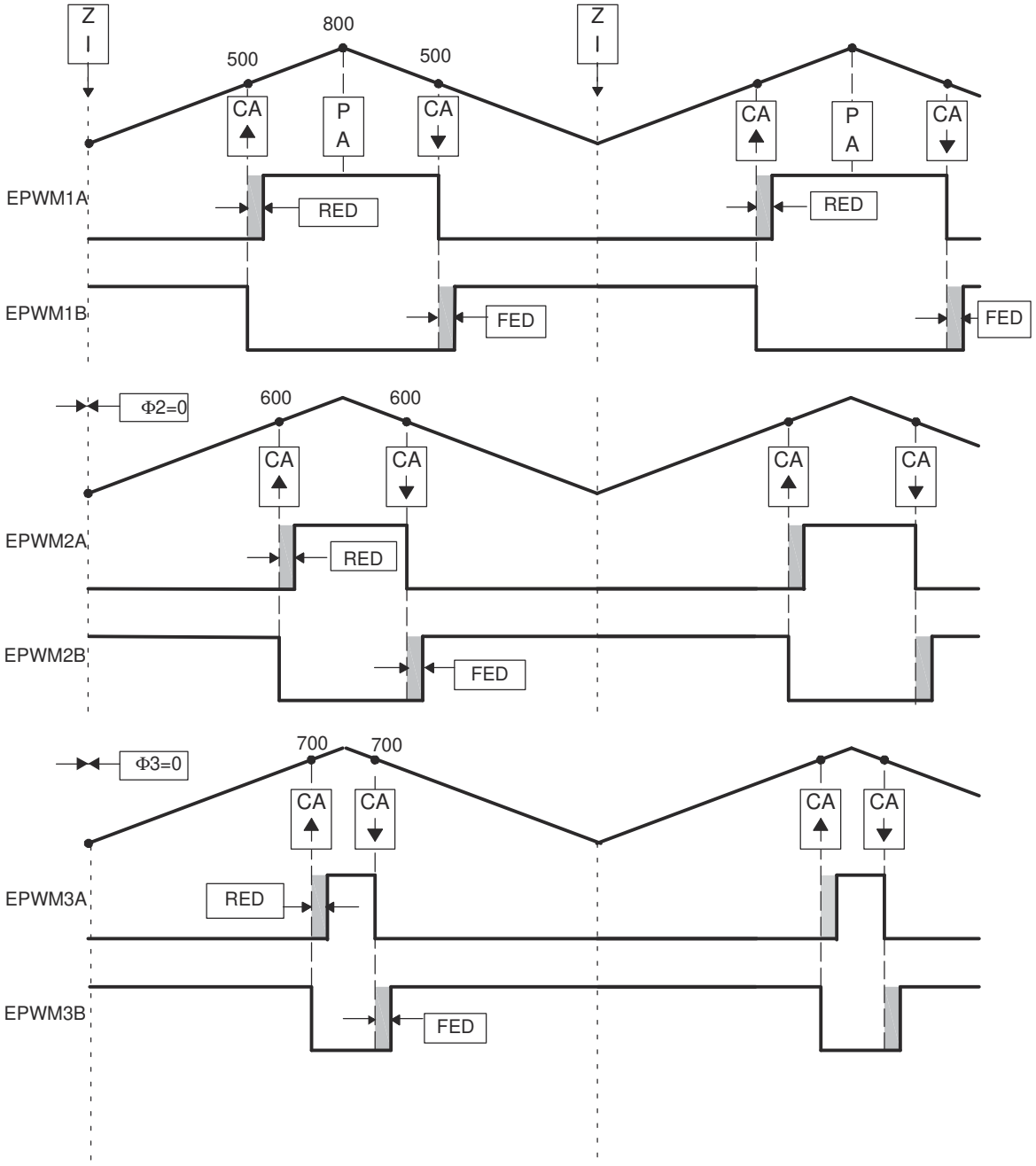



**Figure 15-68. 3-Phase Inverter Waveforms for Control of Dual 3-Phase Inverter Stages (Only One Inverter Shown)**


Copyright © 2024 Texas Instruments Incorporated

### 15.13.7 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 15-69 shows a master and slave module with a phase relationship of 120° (that is, the slave leads the master).

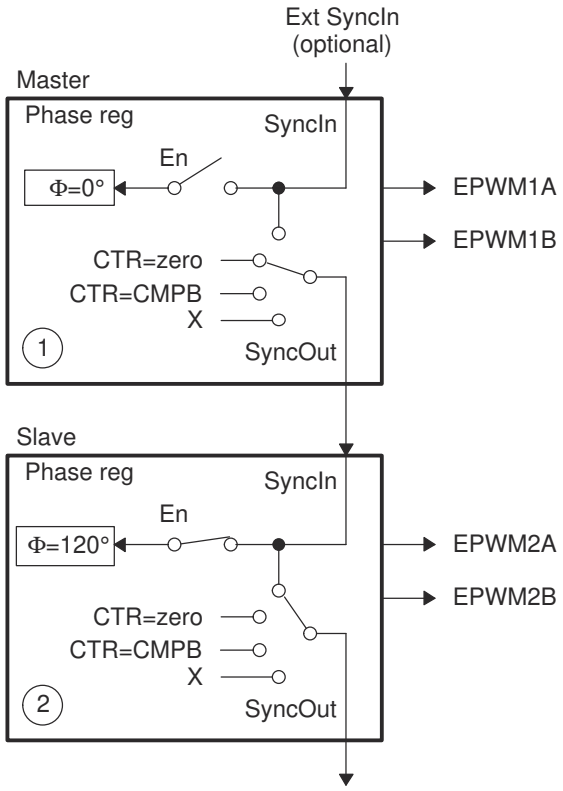



**Figure 15-69. Configuring Two PWM Modules for Phase Control**

Figure 15-70 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 x 360° = 120°). 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°.



Copyright © 2024 Texas Instruments Incorporated

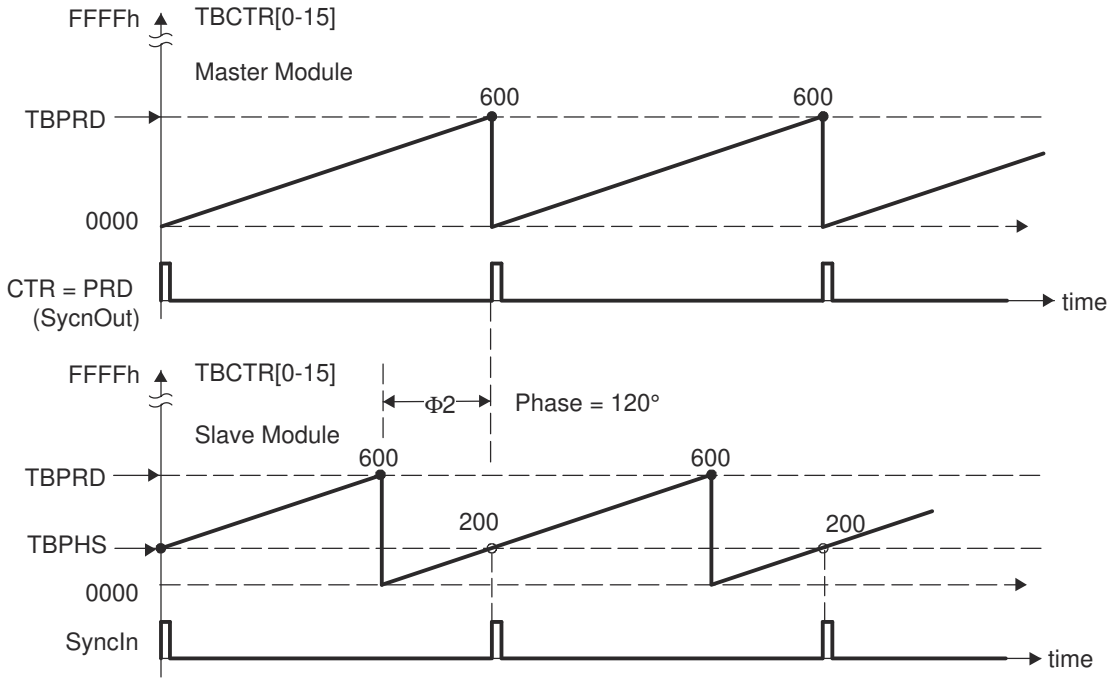



**Figure 15-70. Timing Waveforms Associated with Phase Control Between Two Modules**

### *15.13.8 Controlling a 3-Phase Interleaved DC/DC Converter*

A popular power topology that makes use of phase-offset between modules is shown in Figure 15-71. This system uses three PWM modules, with module 1 configured as the master. To work, the phase relationship between adjacent modules must be F = 120°. 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) x (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 15-72 shows the waveforms for the configuration in Figure 15-71.



Copyright © 2024 Texas Instruments Incorporated

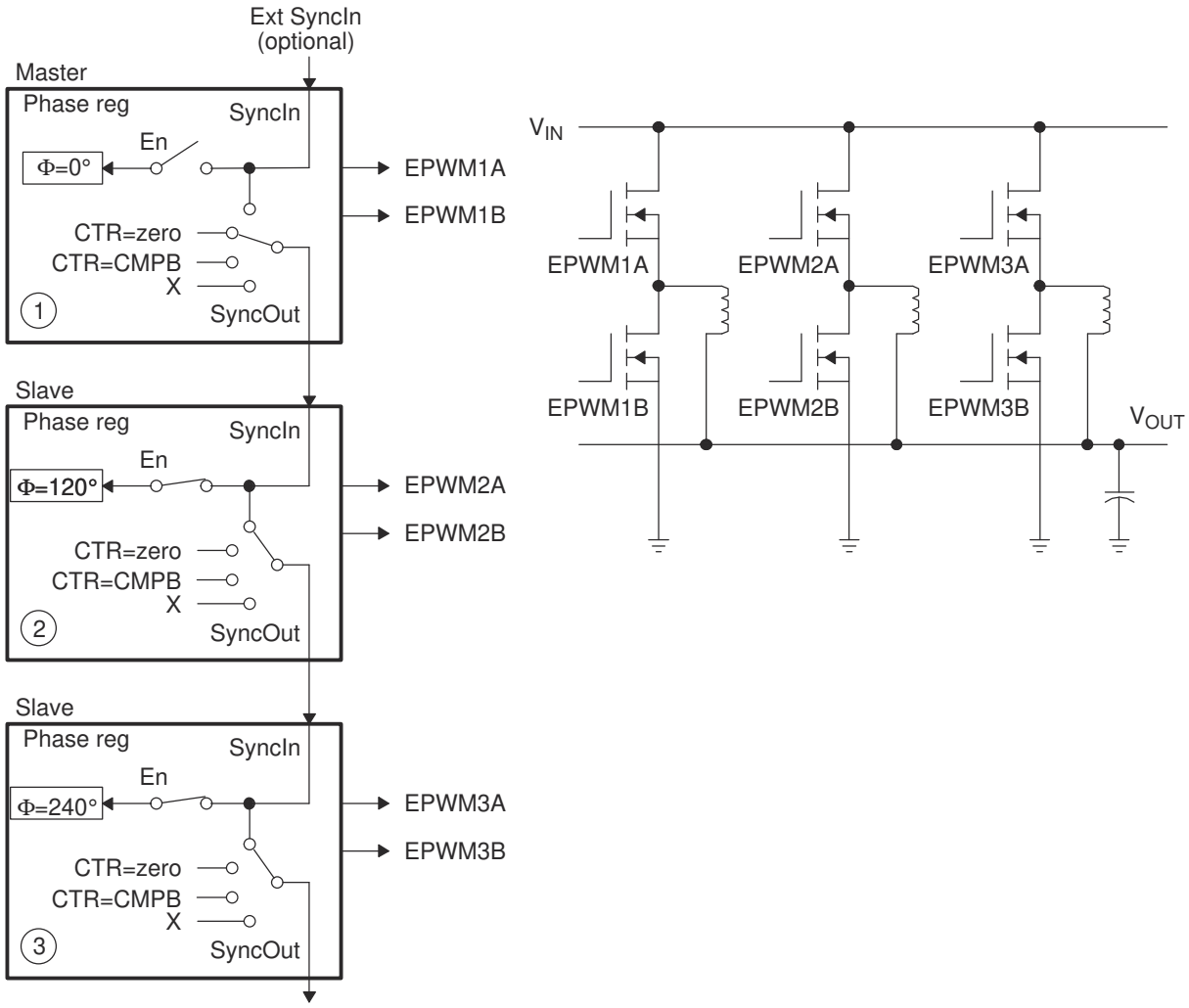



**Figure 15-71. Control of 3-Phase Interleaved DC/DC Converter**



Copyright © 2024 Texas Instruments Incorporated

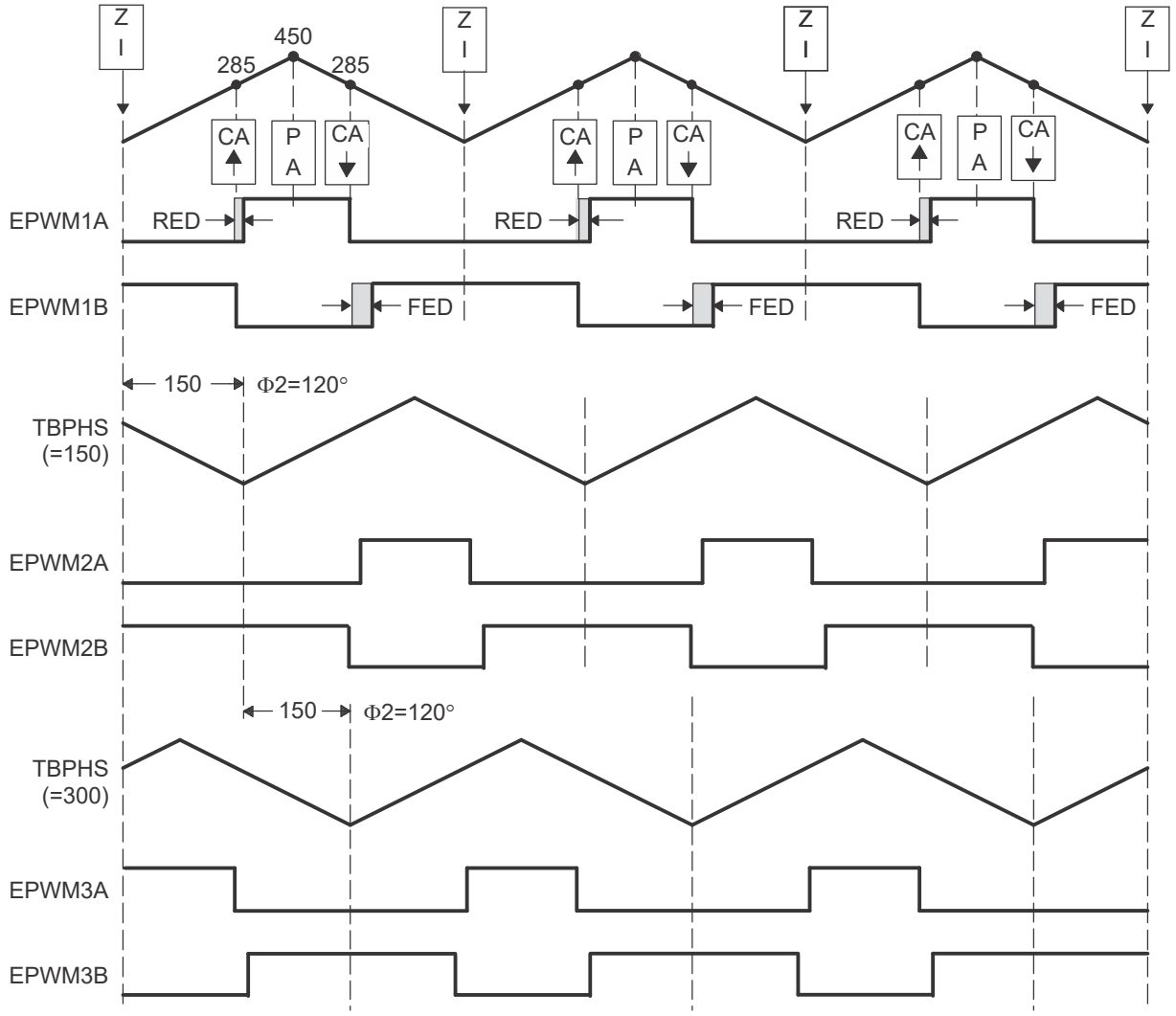



**Figure 15-72. 3-Phase Interleaved DC/DC Converter Waveforms for Control of 3-Phase Interleaved DC/DC Converter**



Copyright © 2024 Texas Instruments Incorporated

### 15.13.9 Controlling Zero Voltage Switched Full Bridge (ZVSFB) Converter

The example given in Figure 15-73 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 15-74 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.

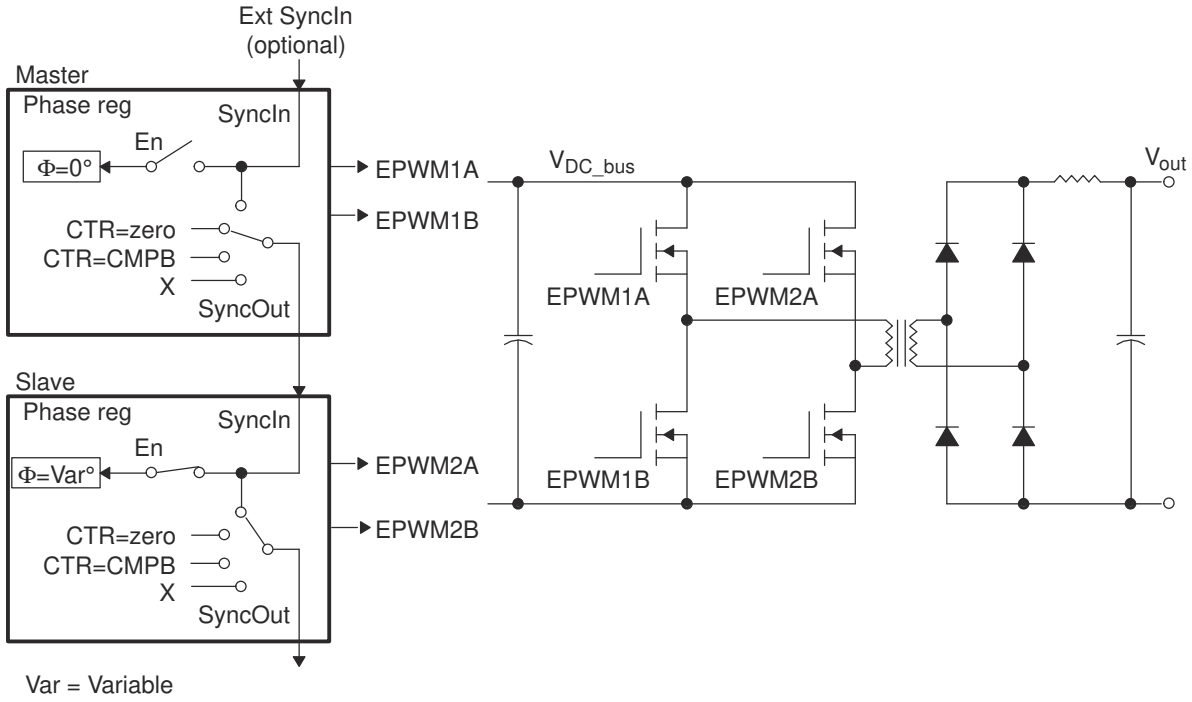



**Figure 15-73. Control of Full-H Bridge Stage ($F_{PWM2}$ = $F_{PWM1}$)**



Copyright © 2024 Texas Instruments Incorporated

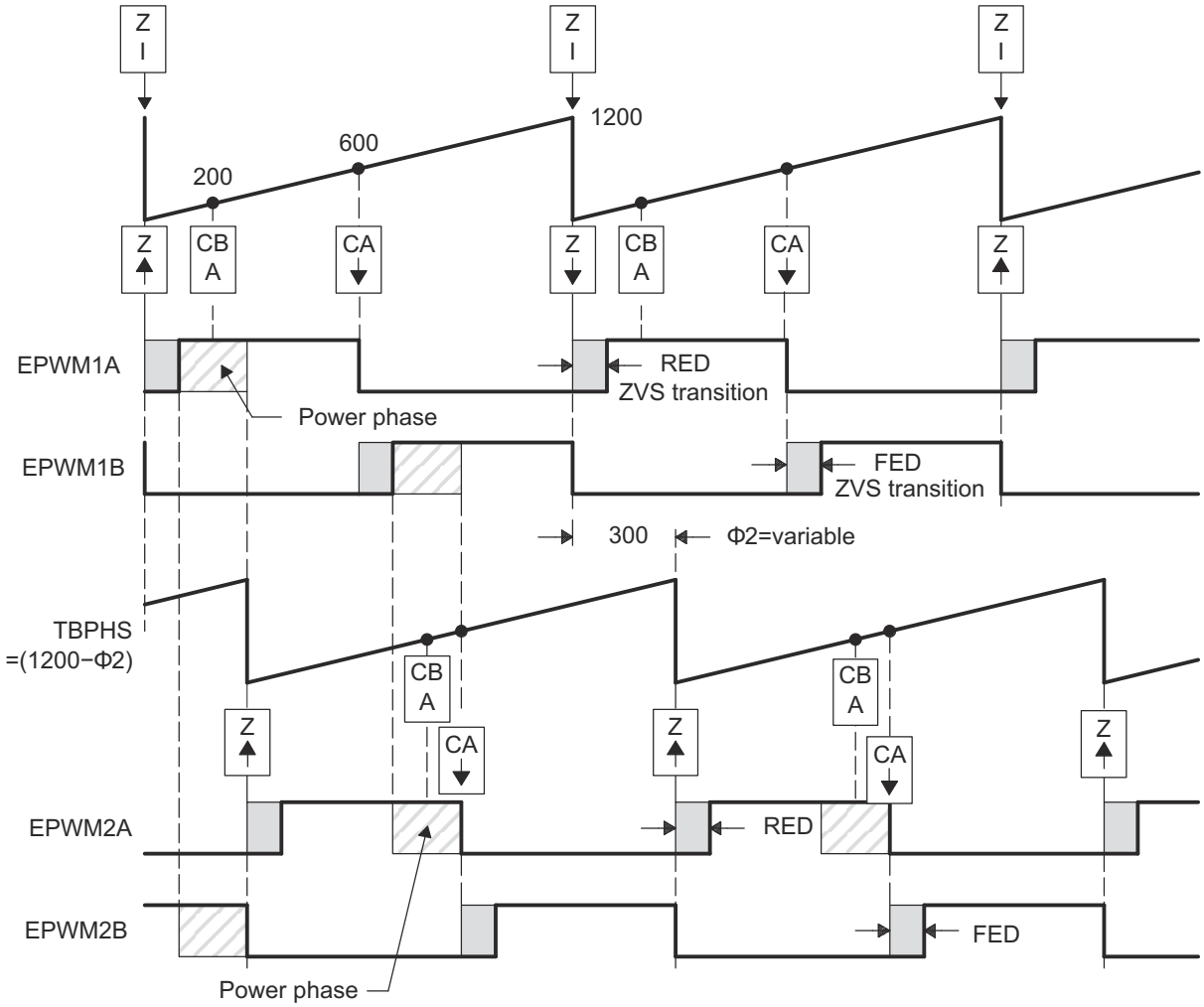



**Figure 15-74. ZVS Full-H Bridge Waveforms**



Copyright © 2024 Texas Instruments Incorporated

### 15.13.10 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 15-75 shows the use of ePWM1A along with the on-chip analog comparator for buck converter topology. The output current is sensed through a current sense resistor and fed to the positive terminal of the on-chip comparator. The internal programmable 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 15-76 shows the waveforms generated by the configuration.

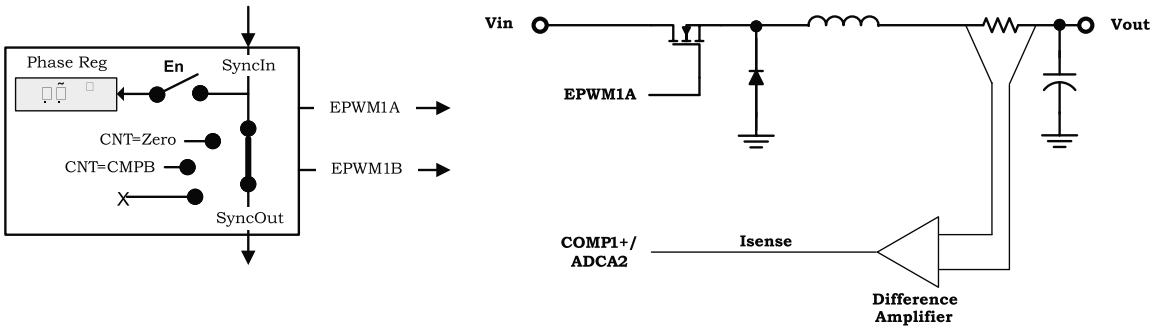



**Figure 15-75. Peak Current Mode Control of Buck Converter**

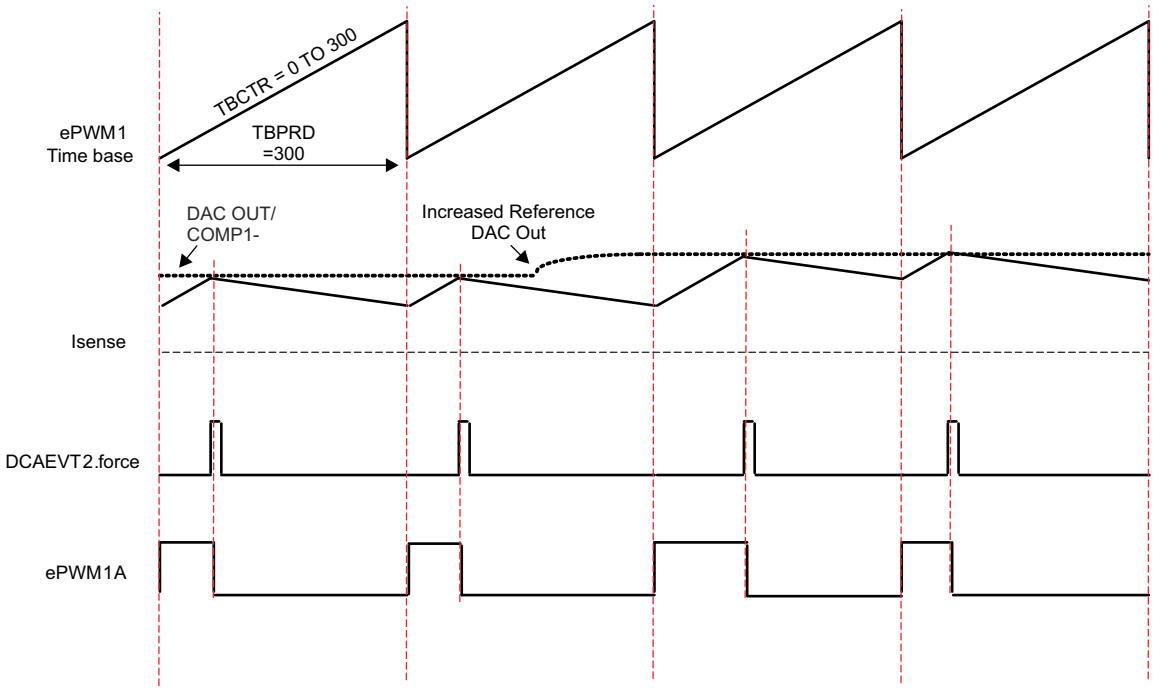


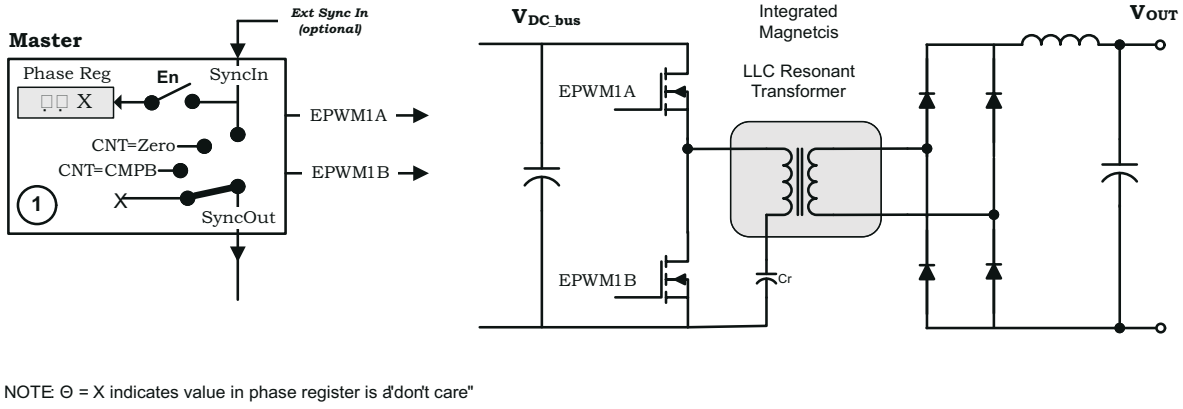

**Figure 15-76. Peak Current Mode Control Waveforms for Control of Buck Converter**



Copyright © 2024 Texas Instruments Incorporated

### 15.13.11 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: Θ = X indicates value in phase register is a "don't care"

**Figure 15-77. Control of Two Resonant Converter Stages**

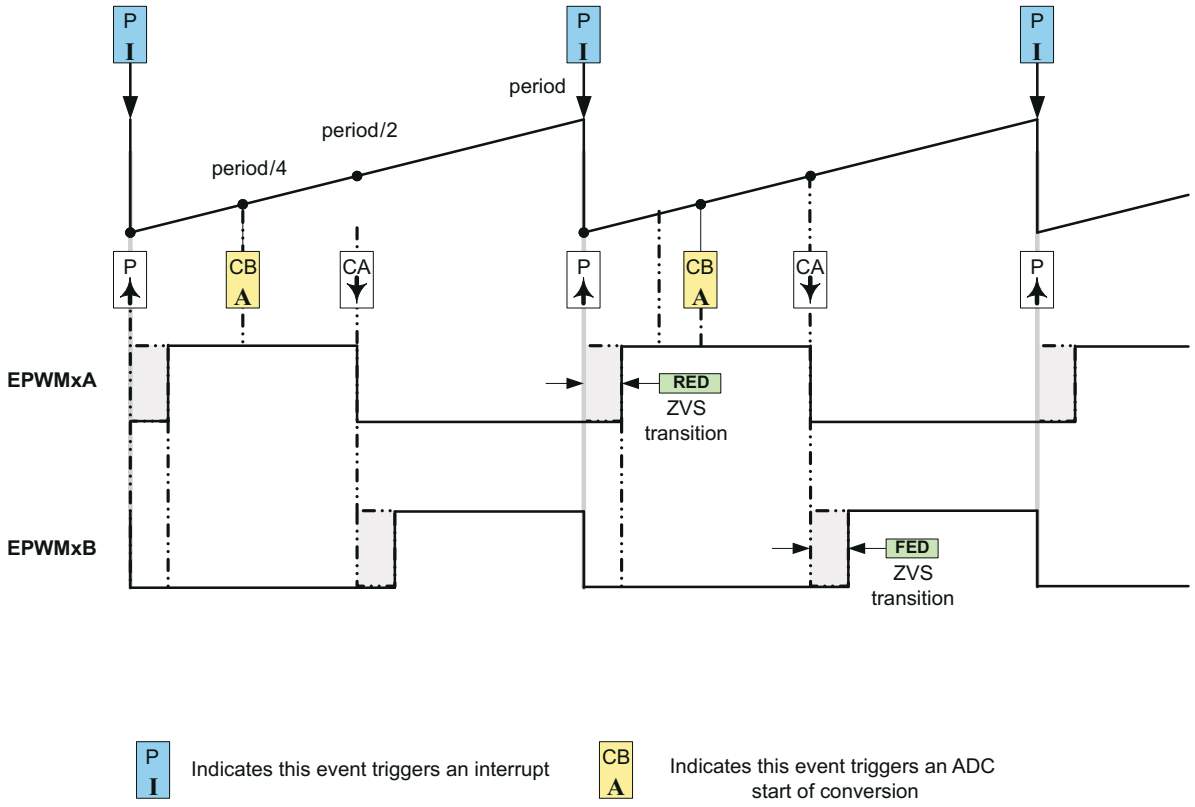



**Figure 15-78. H-Bridge LLC Resonant Converter PWM Waveforms**



Copyright © 2024 Texas Instruments Incorporated

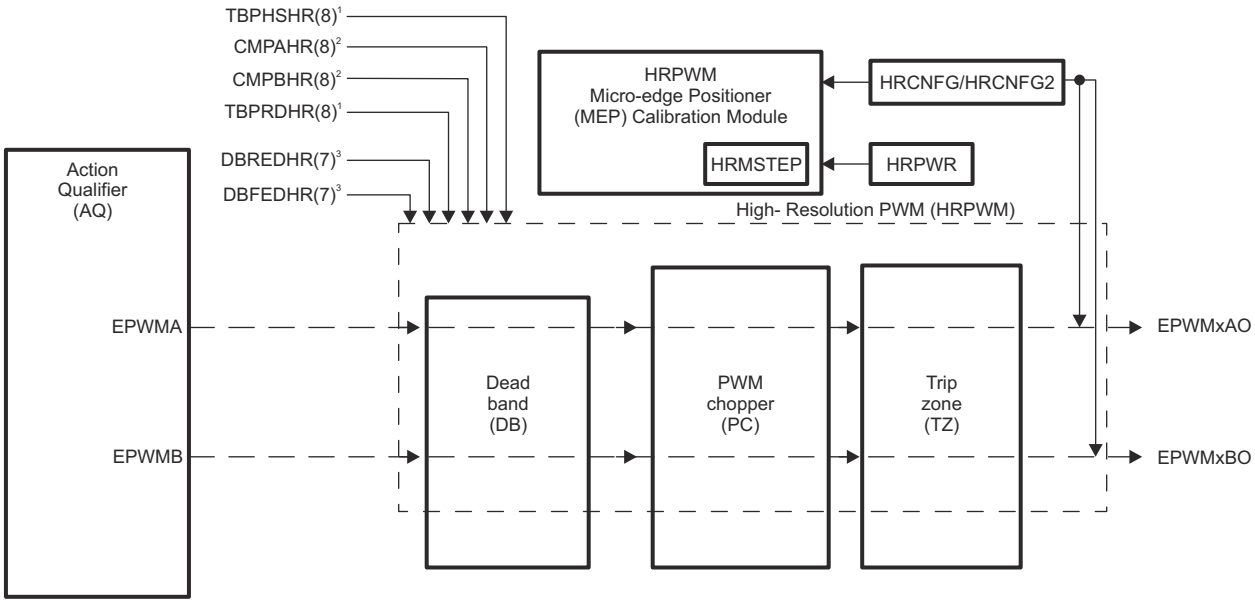
## 15.14 High-Resolution Pulse Width Modulator (HRPWM)

Figure 15-79 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 15-79. HRPWM Block Diagram**



Copyright © 2024 Texas Instruments Incorporated

The ePWM peripheral is used to perform a function mathematically equivalent to a digital-to-analog converter (DAC). As shown in Figure 15-80, the effective resolution for conventionally generated PWM is a function of PWM frequency (or period) and system clock frequency.

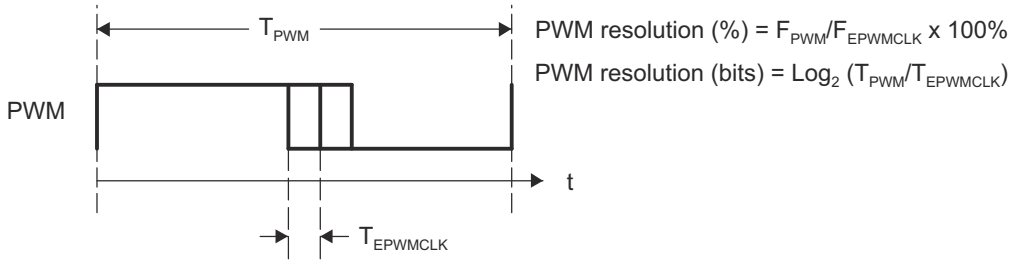



**Figure 15-80. Resolution Calculations for Conventionally Generated PWM**

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 15-13 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 15-13. 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



Copyright © 2024 Texas Instruments Incorporated

### 15.14.1 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 15.14.1.7.

Figure 15-81 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 15.14.1.8.

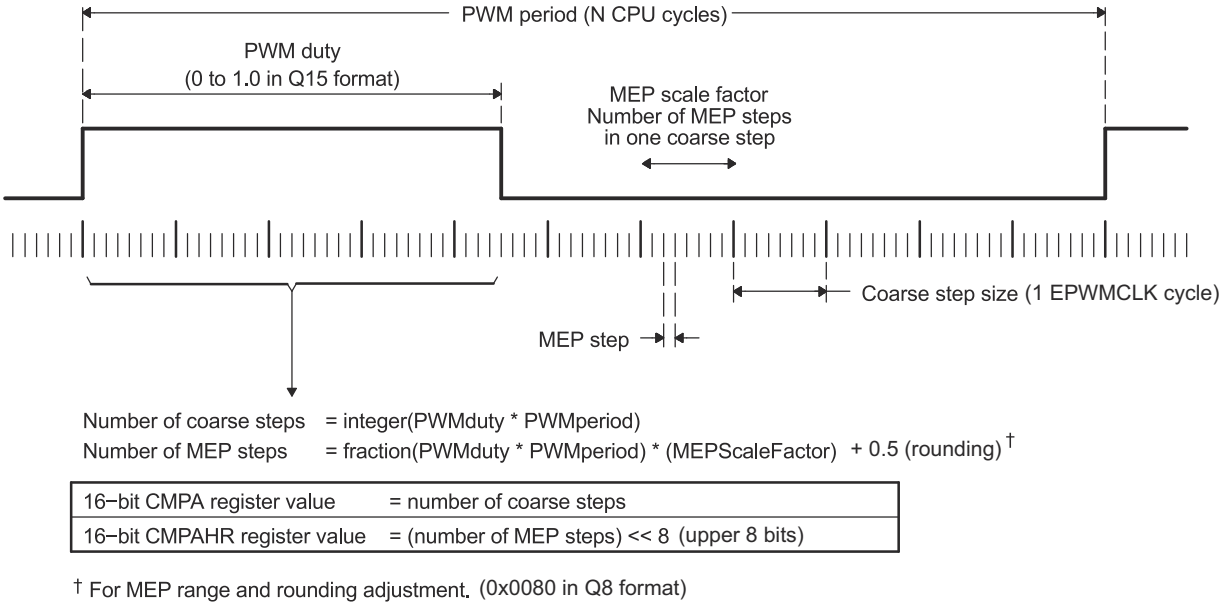



**Figure 15-81. Operating Logic Using MEP**

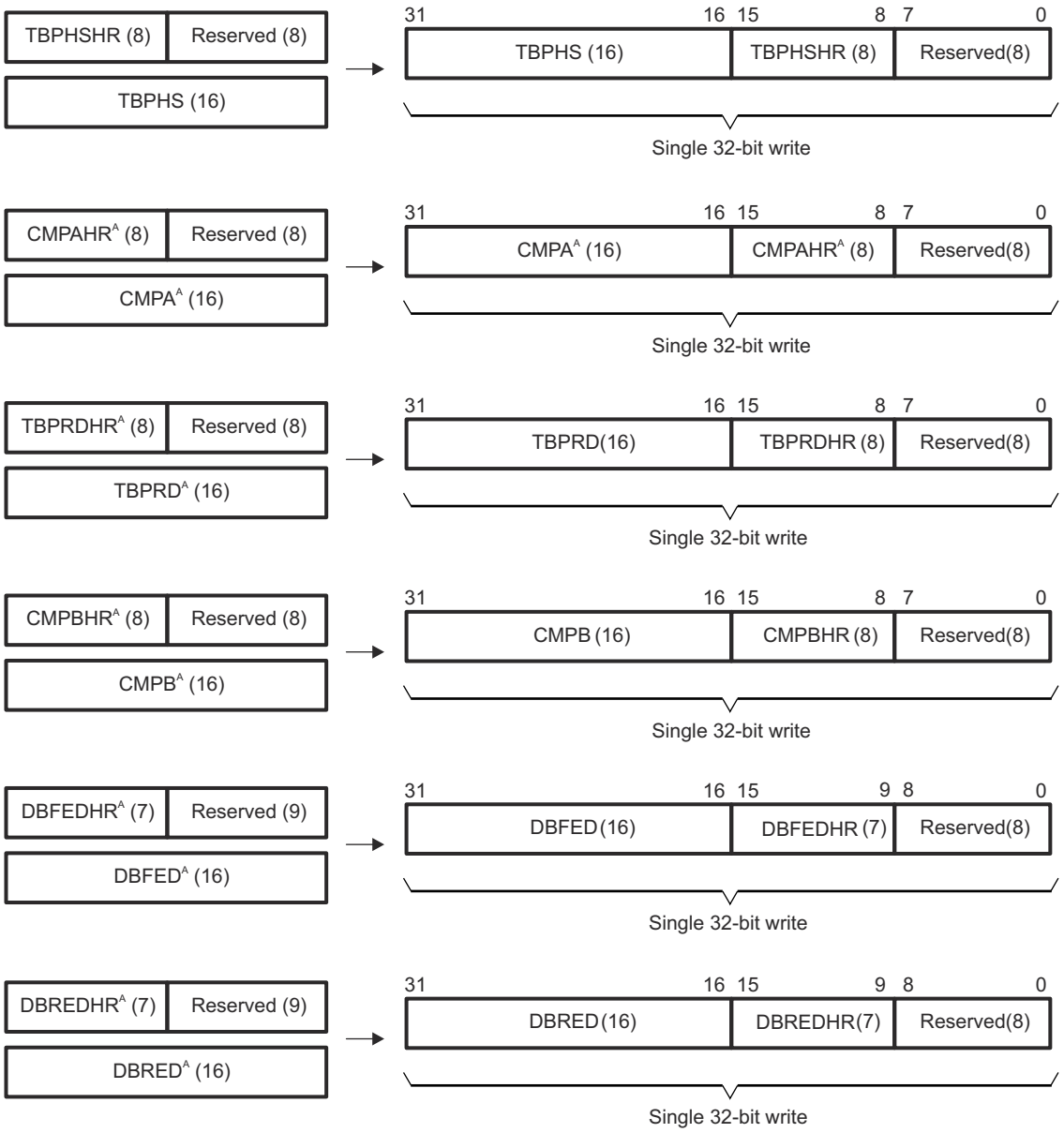
#### 15.14.1.1 Controlling the HRPWM Capabilities

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



Copyright © 2024 Texas Instruments Incorporated




A. Dependent upon your device, these registers can be mirrored and can be written to at two different memory locations.

**Figure 15-82. HRPWM Extension Registers and Memory Configuration**

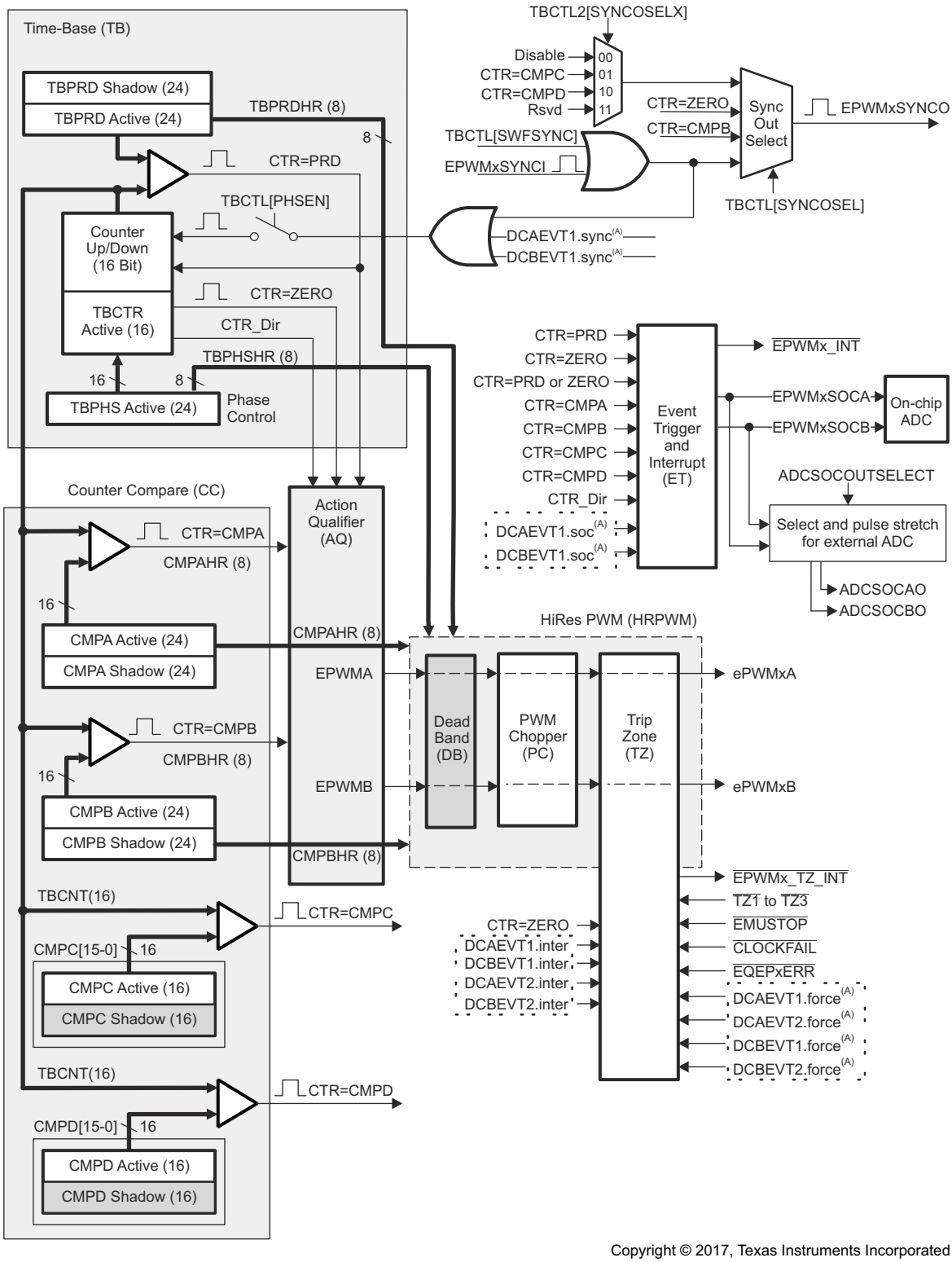
**Note**

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 15-83 shows how the HRPWM interfaces with the 8-bit extension registers.



Copyright © 2024 Texas Instruments Incorporated




Copyright © 2017, Texas Instruments Incorporated

A. These events are generated by the ePWM Digital Compare (DC) submodule based on the levels of the TRIPIN inputs.

**Figure 15-83. HRPWM System Interface**



Copyright © 2024 Texas Instruments Incorporated

#### 15.14.1.2 HRPWM Source Clock

The HRPWM module and HRCAL module are clocked from the EPWM1CLK. Therefore, EPWM1CLK must be enabled for the HRCAL or any of the HRPWM modules to be enabled. Figure 15-84 shows the HRCAL and HRPWM modules sourced from the ePWM1 clock source.

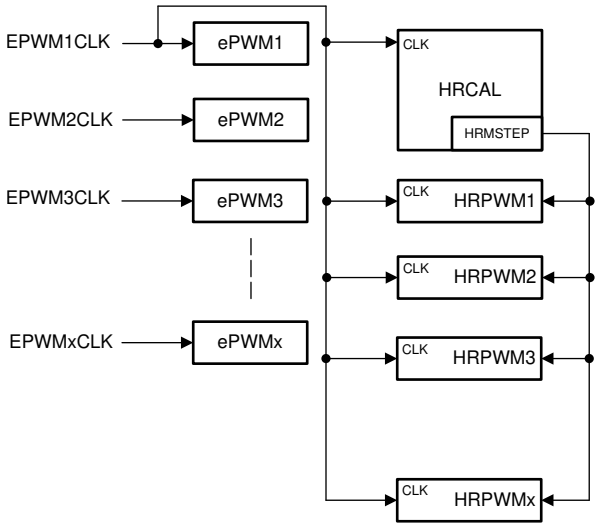



**Figure 15-84. HRPWM and HRCAL Source Clock**

#### 15.14.1.3 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. |



Copyright © 2024 Texas Instruments Incorporated

**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 = fraction(PWMduty*PWMperiod)<<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 = (fraction(PWMduty * PWMperiod) * MEP Scale Factor + 0.5)<<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 = (fraction(PWMduty * PWMperiod) * MEP Scale Factor + 0.5)<<8).

**Note**

HRPWM functionality is only available on CPU1.



Copyright © 2024 Texas Instruments Incorporated

#### 15.14.1.4 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.

#### 15.14.1.5 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 15-14 shows the typical range of operating frequencies supported by the HRPWM.

**Table 15-14. Relationship Between MEP Steps, PWM Frequency, and Resolution**

| System (MHz) | MEP Steps Per EPWMCLK (1) (2) (3) | PWM Minimum (Hz) (4) | PWM Maximum (MHz) | Resolution at Maximum (Bits) (5) |
|---|---|---|---|---|
| 60.0 | 93 | 916 | 3.00 | 10.9 |
| 70.0 | 79 | 1068 | 3.50 | 10.6 |
| 80.0 | 69 | 1221 | 4.00 | 10.4 |
| 90.0 | 62 | 1373 | 4.50 | 10.3 |
| 100.0 | 56 | 1526 | 5.00 | 10.1 |

(1) TBCLK = EPWMCLK.
(2) Table data based on a MEP time resolution of 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.



Copyright © 2024 Texas Instruments Incorporated

#### 15.14.1.5.1 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 15-85, a compare value of 32 counts (duty = 40%) is the closest to 40.5% that can be attained. This is equivalent to an edge position of 320ns instead of the desired 324ns. This data is shown in Table 15-15.

By utilizing the MEP, an edge position much closer to the desired point of 324ns can be achieved. Table 15-15 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.

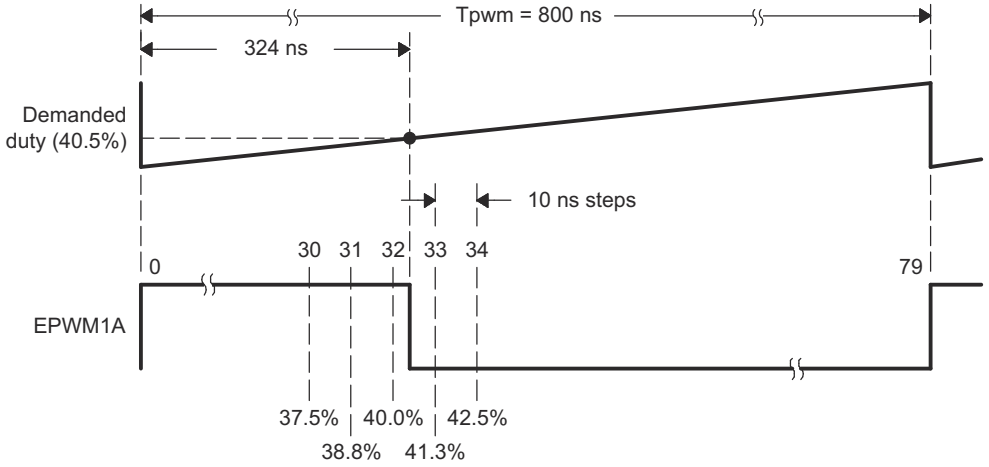



**Figure 15-85. Required PWM Waveform for a Requested Duty = 40.5%**

**Table 15-15. CMPA versus Duty (left), and [CMPA:CMPAHR] versus Duty (right)**

| CMPA (count)(1) (2) (3) | Duty (%) | High Time (ns) | CMPA (count) | CMPAHR (count) | Duty (%) | High Time (ns) |
|---|---|---|---|---|---|---|
| 28 | ***35.0*** | 280 | 32 | 18 | ***40.405*** | 323.24 |
| 29 | ***36.3*** | 290 | 32 | 19 | ***40.428*** | 323.42 |
| 30 | ***37.5*** | 300 | 32 | 20 | ***40.450*** | 323.60 |
| 31 | ***38.8*** | 310 | 32 | 21 | ***40.473*** | 323.78 |
| 32 | ***40.0*** | 320 | 32 | 22 | ***40.495*** | 323.96 |
| 33 | ***41.3*** | 330 | 32 | 23 | ***40.518*** | 324.14 |
| 34 | ***42.5*** | 340 | 32 | 24 | ***40.540*** | 324.32 |
| | | | 32 | 25 | ***40.563*** | 324.50 |
| Required | | | 32 | 26 | ***40.585*** | 324.68 |
| 32.40 | ***40.5*** | 324 | 32 | 27 | ***40.608*** | 324.86 |

(1) 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



Copyright © 2024 Texas Instruments Incorporated

#### 15.14.1.5.2 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, **PWMDuty** | = | 0.405 (40.5%) |
| PWM period in terms of coarse steps, **PWMPeriod** (800ns/10ns) | = | 80 |
| Number of MEP steps per coarse step at 180ps (10ns/180ps), **MEP_ScaleFactor** | = | 55 |
| Value to keep CMPAHR within the range of 1-255 and fractional rounding constant (default value) | = | 0.5 (0080h in Q8 format) |

**Step 1: Percentage Integer Duty value conversion for CMPA register**

| | | |
|---|---|---|
| CMPA register value | = | int(**PWMDuty***PWMPeriod**); int means integer part |
| | = | int(0.405 * 80) |
| | = | int(32.4) |
| CMPA register value | = | 32 (20h) |

**Step 2: Fractional value conversion for CMPAHR register**

| | | |
|---|---|---|
| CMPAHR | = | (frac(**PWMDuty***PWMPeriod**)*MEP_ScaleFactor** + 0.5) <<8); frac means fractional part |
| | = | (frac(32.4) * 55 + 0.5) <<8; Shifting is to move the value to the high byte of CMPAHR. |
| | = | (0.4 * 55 + 0.5) <<8 |
| | = | (22 + 0.5) <<8 |
| | = | 22.5 * 256; Shifting left by 8 is the same as multiplying by 256. |
| | = | 5760 (1680h) |
| CMPAHR | = | 1680h CMPAHR value = 1600h (lower 8 bits are ignored by hardware). |



Copyright © 2024 Texas Instruments Incorporated

**Note**

If the AUTOCONV bit (HRCNFG.6) is set and the MEP_ScaleFactor is in the HRMSTEP register, then CMPAHR / CMPBHR register value = frac (PWMDuty*PWMperiod<<8). The rest of the conversion calculations are performed automatically in hardware, and the correct MEP-scaled signal edge appears on the ePWM channel output. If AUTOCONV is not set, the above calculations must be performed by software.

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 15.14.1.8. The actual implementation takes advantage of the 32-bit CPU architecture and is somewhat different from the steps shown in Section 15.14.1.5.2.

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.

#### *15.14.1.5.3 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 15-86 to Figure 15-89. This limitation imposes a duty cycle limit on the MEP. For example, precision edge control is not available all the way down to 0% duty cycle. When high-resolution period control is disabled, regular PWM duty control is fully operational down to 0% duty cycle despite the unavailability of HRPWM features in the first three cycles. In most applications, this 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 15-16. 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.



Copyright © 2024 Texas Instruments Incorporated

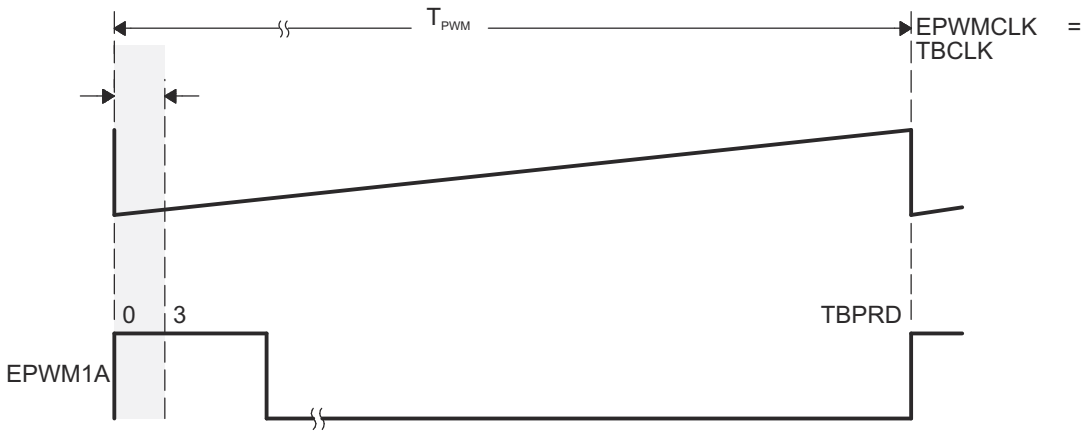



**Figure 15-86. Low % Duty Cycle Range Limitation Example (HRPCTL[HRPE] = 0)**

**Table 15-16. Duty Cycle Range Limitation for Three EPWMCLK/TBCLK Cycles**

| PWM Frequency(1) (kHz) | 3 Cycles Minimum Duty | 3 Cycles Maximum Duty(2) |
|---|---|---|
| 200 | 0.6% | 99.4% |
| 400 | 1.2% | 98.8% |
| 600 | 1.8% | 98.2% |
| 800 | 2.4% | 97.6% |
| 1000 | 3% | 97% |
| 1200 | 3.6% | 96.4% |
| 1400 | 4.2% | 95.8% |
| 1600 | 4.8% | 95.2% |
| 1800 | 5.4% | 94.6% |
| 2000 | 6% | 94% |

(1) EPWMCLK = TBCLK = 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 15-87. 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 15-16.

**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.



Copyright © 2024 Texas Instruments Incorporated

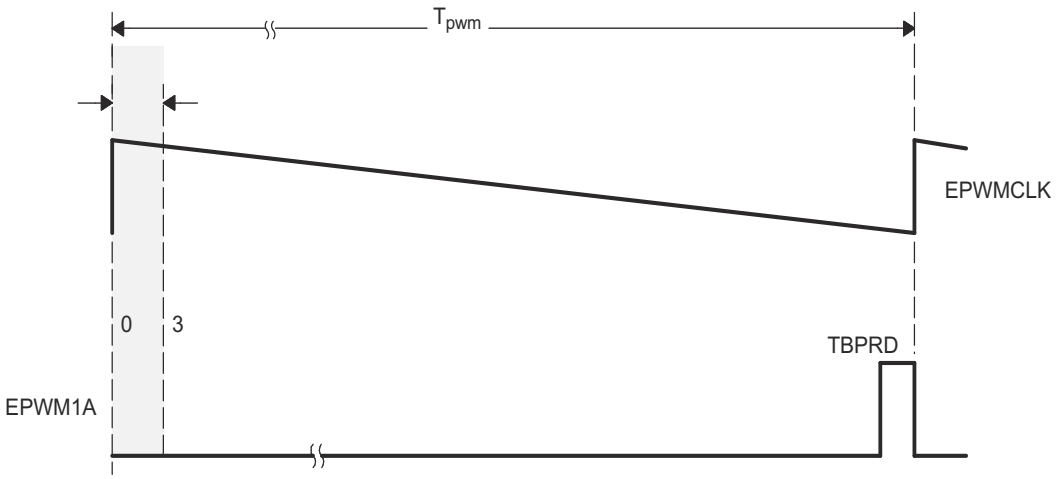



**Figure 15-87. High % Duty Cycle Range Limitation Example (HRPCTL[HRPE] = 0)**

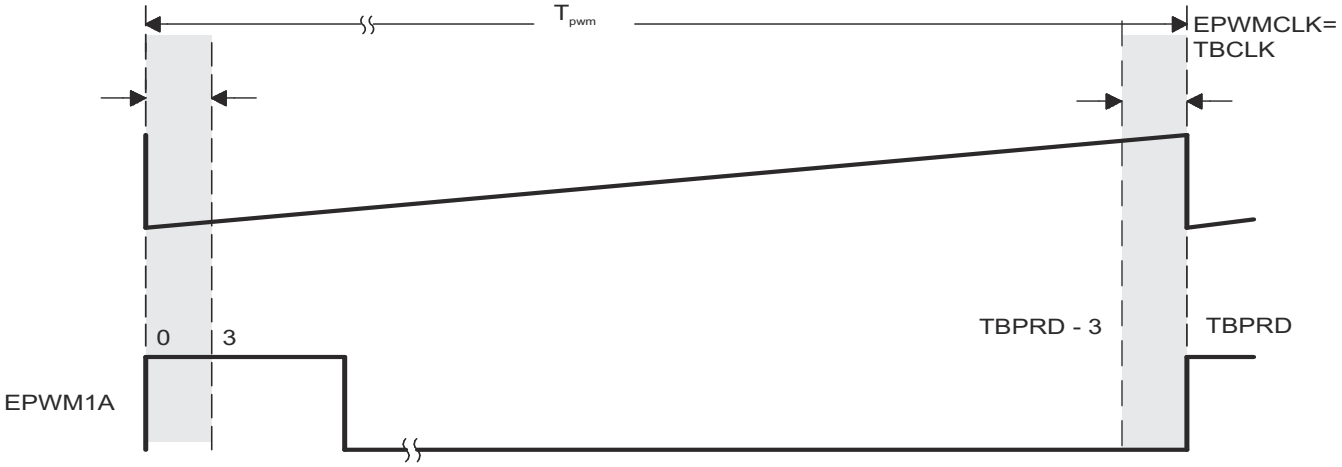



**Figure 15-88. Up-Count Duty Cycle Range Limitation Example (HRPCTL[HRPE] = 1)**

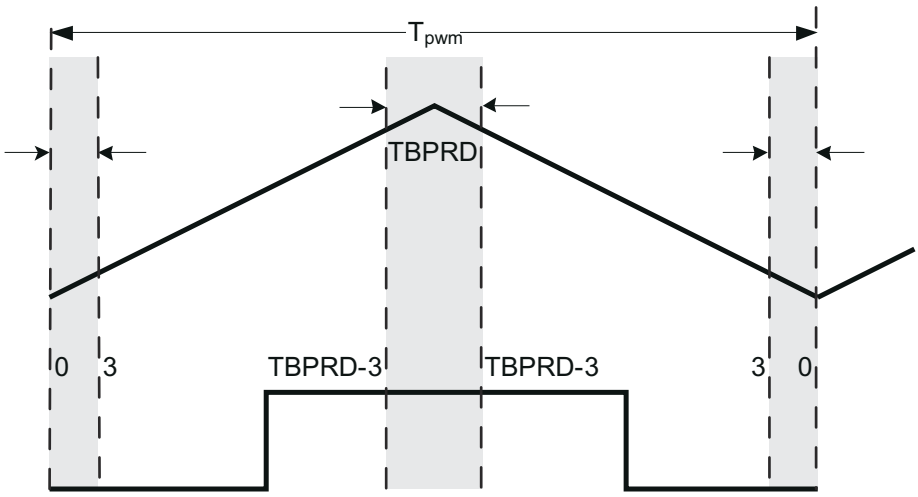



**Figure 15-89. Up-Down Count Duty Cycle Range Limitation Example (HRPCTL[HRPE] = 1)**



Copyright © 2024 Texas Instruments Incorporated

#### 15.14.1.5.4 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 ±1 TBCLK cycle jitter in up-count mode and ±2 TBCLK cycle jitter in up-down count mode.

The scaling procedure described for duty cycle in Section 15.14.1.5.2 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**

| | |
|---|---|
| Integer period value | = 571 * $T_{TBCLK}$ |
| | = int (571.428) * $T_{TBCLK}$ |
| | = int (PWMperiod) * $T_{TBCLK}$ |
| In up-count mode: | |
| TBPRD | = 570 (TBPRD = period value - 1) |
| | = 023Ah |
| In up-down count mode: | = 285 (TBPRD = period value/2) |
| TBPRD | = 011Dh |



Copyright © 2024 Texas Instruments Incorporated

**Step 2: Fractional value conversion for TBPRDHR register**

In up-count mode:

| | |
|---|---|
| TBPRDHR register value | = (frac(PWMperiod) * MEP_ScaleFactor + 0.5) |
| If auto-conversion enabled and HRMSTEP = | |
| MEP_ScaleFactor value (55): | = frac (PWMperiod) << 8 (Shifting is to move the value to the high byte of TBPRDHR) |
| TBPRDHR register value | = frac (571.428) << 8 |
| | = 0.428 × 256 |
| | = 6D00h |
| The auto-conversion then automatically performs the calculation, such that TBPRDHR MEP delay is scaled by hardware to: | = ((TBPRDHR(15:0) >> 8) × HRMSTEP + 80h) << 8 |
| | = (006Dh × 55 + 80h) >> 8 |
| | = (17EBh) >> 8 |
| Period MEP delay | = 0017h MEP Steps |

In up-down count mode:

| | |
|---|---|
| TBPRDHR register value | = (frac(PWMperiod) * MEP_ScaleFactor + 0.5) |
| If auto-conversion enabled and HRMSTEP = | |
| MEP_ScaleFactor value (55): | = frac (PWMperiod / 2) << 8 (Shifting is to move the value to the high byte of TBPRDHR) |
| TBPRDHR register value | = frac (285.714) << 8 |
| | = 0.714 × 256 |
| | = B600h |
| The auto-conversion then automatically performs the calculation, such that TBPRDHR MEP delay is scaled by hardware to: | = ((TBPRDHR(15:0) >> 8) × HRMSTEP + 80h) << 8 |
| | = (00B6h × 55 + 80h) >> 8 |
| | = (279Ah) >> 8 |
| Period MEP delay | = 0027h MEP Steps |



Copyright © 2024 Texas Instruments Incorporated

#### 15.14.1.5.4.1 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 15.14.2.
11. To control high-resolution period, write to the TBPRDHR(M) registers.

**Note**

When high-resolution period mode is enabled, an EPWMxSYNC pulse introduces ±1-2 cycle jitter to the PWM (±1 cycle in up-count mode and ±2 cycle in up-down count mode). For this reason, TBCTL[SYNCOSEL] cannot be set to 1 (CTR = 0 is EPWMxSYNCO source) or 2 (CTR = CMPB is EPWMxSYNCO source). Otherwise, the jitter occurs on every PWM cycle with the synchronization pulse.

When TBCTL[SYNCOSEL] = 0 (EPWMxSYNCI 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.



Copyright © 2024 Texas Instruments Incorporated

#### 15.14.1.6 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 Required = 3.75ns

With 55 MEP steps per coarse step at 180ps each:



Copyright © 2024 Texas Instruments Incorporated

**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.

#### 15.14.1.7 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.



Copyright © 2024 Texas Instruments Incorporated

#### 15.14.1.8 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 15-2 assumes MEP step size of 150ps and does not use the SFO library.

***Example 15-2. #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
```



Copyright © 2024 Texas Instruments Incorporated

#### 15.14.1.8.1 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 15-90 and Figure 15-91 show the required PWM waveform. As explained previously, configuration for the ePWM1 module is almost identical to the normal case except that the appropriate MEP options need to be enabled/selected.

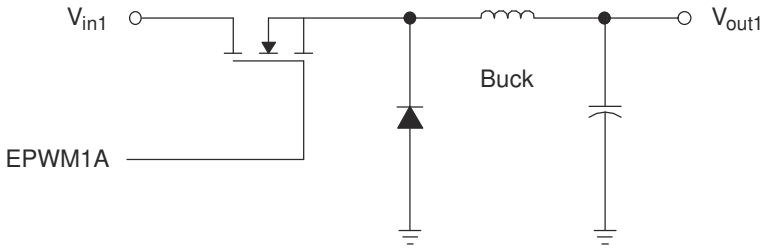



**Figure 15-90. Simple Buck Controlled Converter Using a Single PWM**

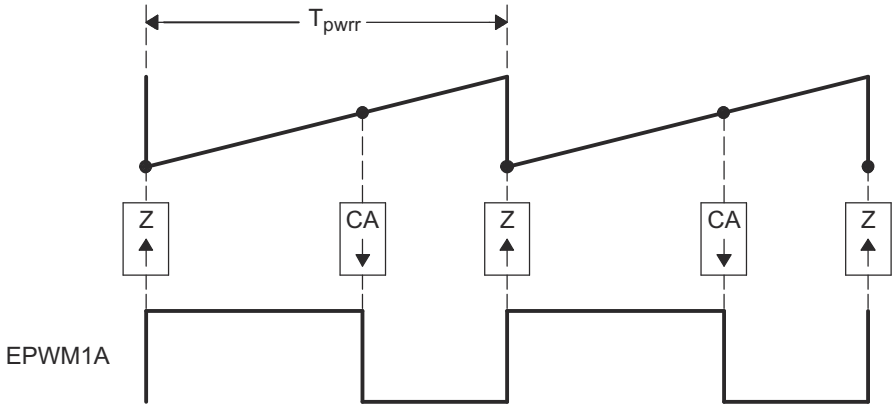



**Figure 15-91. PWM Waveform Generated for Simple Buck Controlled Converter**

The example code shown consists of two main parts:

- Initialization code (executed once)
- Run time code (typically executed within an ISR)

Example 15-3 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 15-4 shows an assembly example of run-time code for the HRPWM buck converter.



Copyright © 2024 Texas Instruments Incorporated

### Example 15-3. HRPWM Buck Converter Initialization Code

```
void HrBuckDrvCnf(void)
 {
// Config for conventional PWM first
EPwm1Regs.TBCTL.bit.PRDLD = 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.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;
EPwm1Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;
EPwm1Regs.TBCTL.bit.CLKDIV = TB_DIV1;
// Note: ChB is initialized here only for comparison purposes, it is not required

EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;
EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;     // optional
EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;       // optional
EPwm1Regs.AQCTLA.bit.ZRO = AQ_SET;
EPwm1Regs.AQCTLA.bit.CAU = AQ_CLEAR;
EPwm1Regs.AQCTLB.bit.ZRO = AQ_SET;                // optional
EPwm1Regs.AQCTLB.bit.CBU = AQ_CLEAR;              // optional
// Now configure the HRPWM resources
EALLOW;                                           // Note these registers are protected
                                                  // and act only on ChA
EPwm1Regs.HRCNFG.all = 0x0;                       // clear all bits first
EPwm1Regs.HRCNFG.bit.EDGMODE = HR_FEP;            // Control Falling Edge Position
EPwm1Regs.HRCNFG.bit.CTLMODE = HR_CMP;            // CMPAHR controls the MEP
EPwm1Regs.HRCNFG.bit.HRLOAD = HR_CTR_ZERO;        // Shadow load on CTR=Zero
EDIS;
MEP_ScaleFactor = 66*256;                         // Start with typical Scale Factor
                                                  // value for 100MHz
                                                  // Note: Use SFO functions to update
                                                     MEP_ScaleFactor dynamically
}
```

### Example 15-4. 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
```



Copyright © 2024 Texas Instruments Incorporated

#### 15.14.1.8.2 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 15-92 and Figure 15-93 show the DAC function and the required PWM waveform. As explained previously, configuration for the ePWM1 module is almost identical to the normal case except that the appropriate MEP options need to be enabled/selected.

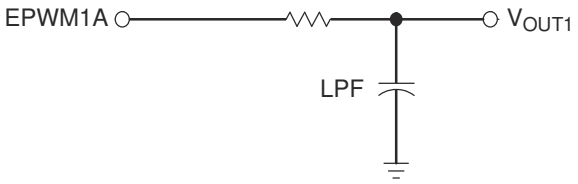



**Figure 15-92. Simple Reconstruction Filter for a PWM-based DAC**

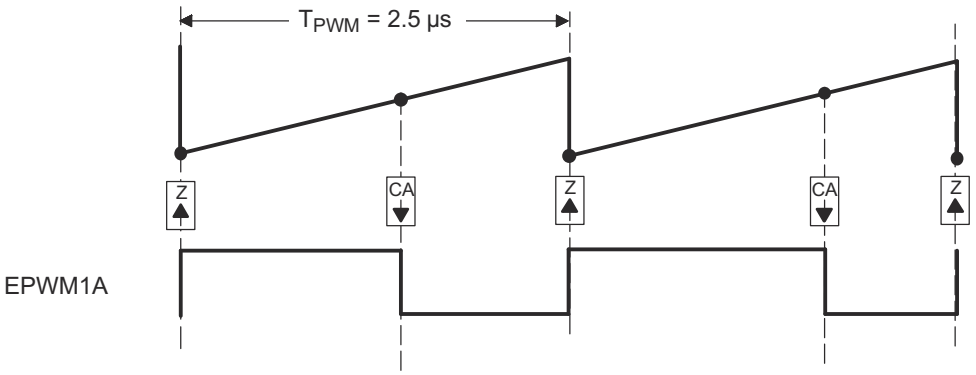



**Figure 15-93. PWM Waveform Generated for the PWM DAC Function**

The example code shown consists of two main parts:

- Initialization code (executed once)
- Run time code (typically executed within an ISR)

This example assumes a typical MEP_SP and does not use the SFO library.

Example 15-5 shows the Initialization code. The first part is configured for conventional PWM. The second part sets up the HRPWM resources.

Example 15-6 shows an assembly example of run-time code that can execute in a high-speed ISR loop.



Copyright © 2024 Texas Instruments Incorporated

### Example 15-5. PWM DAC Function Initialization Code

```
void HrPwmDacDrvCnf(void)
{
// Config for conventional PWM first
EPwm1Regs.TBCTL.bit.PRDLD = 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.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;
EPwm1Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;
EPwm1Regs.TBCTL.bit.CLKDIV = TB_DIV1;
// Note: ChB is initialized here only for comparison purposes, it is not required

EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;
EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;  // optional
EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;    // optional

EPwm1Regs.AQCTLA.bit.ZRO = AQ_SET;
EPwm1Regs.AQCTLA.bit.CAU = AQ_CLEAR;
EPwm1Regs.AQCTLB.bit.ZRO = AQ_SET;             // optional
EPwm1Regs.AQCTLB.bit.CBU = AQ_CLEAR;           // optional
// Now configure the HRPWM resources
EALLOW;                                        // Note these registers are protected
                                               // and act only on ChA.
EPwm1Regs.HRCNFG.all = 0x0;                    // Clear all bits first
EPwm1Regs.HRCNFG.bit.EDGMODE = HR_FEP;         // Control falling edge position
EPwm1Regs.HRCNFG.bit.CTLMODE = HR_CMP;         // CMPAHR controls the MEP.
EPwm1Regs.HRCNFG.bit.HRLOAD = HR_CTR_ZERO;     // Shadow load on CTR=Zero.
EDIS;
MEP_ScaleFactor = 66*256;                      // Start with typical Scale Factor
                                               // value for 100MHz.
                                               // Use SFO functions to update MEP_ScaleFactor
                                               // dynamically.
}
```

### Example 15-6. 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
```



Copyright © 2024 Texas Instruments Incorporated

### 15.14.2 SFO Library Software - SFO_TI_Build_V8.lib

Table 15-17 lists several features of the SFO_TI_Build_V8.lib library.

**Table 15-17. 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 |

#### 15.14.2.1 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 15.14.1.5.3).
- The SFO() function also updates the HRMSTEP register with the scale factor result. If the HRCNFG[AUTOCONV] bit is set, the application software is responsible only for setting CMPAHR = fraction(PWMduty*PWMperiod) << 8 or CMPBHR = fraction(PWMduty*PWMperiod) << 8 or TBPRDHR = fraction (PWMperiod) while running SFO() in the background. The MEP Calibration Module 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 = (fraction(PWMduty * PWMperiod) * MEP Scale Factor) << 8 + 0x080.
  - Similar behavior applies for TBPHSHR, CMPBHR, DBREDHR, and DBFEDHR. Auto-conversion must be enabled when using TBPRDHR.



Copyright © 2024 Texas Instruments Incorporated

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.

#### 15.14.2.2 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 15-18.

**Table 15-18. 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 15-7. 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)
```



Copyright © 2024 Texas Instruments Incorporated

### Step 2. Element Declaration

Declare an integer variable for the scale factor value as shown below.

***Example 15-8. 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 15-9. 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 15-10. 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)
 }
```



Copyright © 2024 Texas Instruments Incorporated

## 15.15 ePWM Registers

This section describes the Enhanced Pulse Width Modulator registers.

### *15.15.1 ePWM Base Addresses*

**Table 15-19. ePWM Base Address Table**

| Device Register | Register Name | Start Address | End Address |
|---|---|---|---|
| EPwm1Regs | EPWM_REGS | 0x0000_4000 | 0x0000_40FF |
| EPwm2Regs | EPWM_REGS | 0x0000_4100 | 0x0000_41FF |
| EPwm3Regs | EPWM_REGS | 0x0000_4200 | 0x0000_42FF |
| EPwm4Regs | EPWM_REGS | 0x0000_4300 | 0x0000_43FF |
| EPwm5Regs | EPWM_REGS | 0x0000_4400 | 0x0000_44FF |
| EPwm6Regs | EPWM_REGS | 0x0000_4500 | 0x0000_45FF |
| EPwm7Regs | EPWM_REGS | 0x0000_4600 | 0x0000_46FF |
| EPwm8Regs | EPWM_REGS | 0x0000_4700 | 0x0000_47FF |
| EPwm9Regs | EPWM_REGS | 0x0000_4800 | 0x0000_48FF |
| EPwm10Regs | EPWM_REGS | 0x0000_4900 | 0x0000_49FF |
| EPwm11Regs | EPWM_REGS | 0x0000_4A00 | 0x0000_4AFF |
| EPwm12Regs | EPWM_REGS | 0x0000_4B00 | 0x0000_4BFF |
| EPwmXbarRegs(1) | EPWM_XBAR_REGS | 0x0000_7A00 | 0x0000_7A3F |

(1) Only available on CPU1.



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2 EPWM_REGS Registers

Table 15-20 lists the memory-mapped registers for the EPWM_REGS registers. All register offset addresses not listed in Table 15-20 should be considered as reserved locations and the register contents should not be modified.

**Table 15-20. EPWM_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | TBCTL | Time Base Control Register | | Go |
| 1h | TBCTL2 | Time Base Control Register 2 | | Go |
| 4h | TBCTR | Time Base Counter Register | | Go |
| 5h | TBSTS | Time Base Status Register | | Go |
| 8h | CMPCTL | Counter Compare Control Register | | Go |
| 9h | CMPCTL2 | Counter Compare Control Register 2 | | Go |
| Ch | DBCTL | Dead-Band Generator Control Register | | Go |
| Dh | DBCTL2 | Dead-Band Generator Control Register 2 | | Go |
| 10h | AQCTL | Action Qualifier Control Register | | Go |
| 11h | AQTSRCSEL | Action Qualifier Trigger Event Source Select Register | | Go |
| 14h | PCCTL | PWM Chopper Control Register | | Go |
| 18h | VCAPCTL | Valley Capture Control Register | | Go |
| 19h | VCNTCFG | Valley Counter Config Register | | Go |
| 20h | HRCNFG | HRPWM Configuration Register | EALLOW | Go |
| 21h | HRPWR | HRPWM Power Register | EALLOW | Go |
| 26h | HRMSTEP | HRPWM MEP Step Register | EALLOW | Go |
| 27h | HRCNFG2 | HRPWM Configuration 2 Register | EALLOW | Go |
| 2Dh | HRPCTL | High Resolution Period Control Register | EALLOW | Go |
| 2Eh | TRREM | HRPWM High Resolution Remainder Register | EALLOW | Go |
| 34h | GLDCTL | Global PWM Load Control Register | EALLOW | Go |
| 35h | GLDCFG | Global PWM Load Config Register | EALLOW | Go |
| 38h | EPWMXLINK | EPWMx Link Register | | Go |
| 40h | AQCTLA | Action Qualifier Control Register For Output A | | Go |
| 41h | AQCTLA2 | Additional Action Qualifier Control Register For Output A | | Go |
| 42h | AQCTLB | Action Qualifier Control Register For Output B | | Go |
| 43h | AQCTLB2 | Additional Action Qualifier Control Register For Output B | | Go |
| 47h | AQSFRC | Action Qualifier Software Force Register | | Go |
| 49h | AQCSFRC | Action Qualifier Continuous S/W Force Register | | Go |
| 50h | DBREDHR | Dead-Band Generator Rising Edge Delay High Resolution Register | | Go |
| 51h | DBRED | Dead-Band Generator Rising Edge Delay Count Register | | Go |
| 52h | DBFEDHR | Dead-Band Generator Falling Edge Delay High Resolution Register | | Go |
| 53h | DBFED | Dead-Band Generator Falling Edge Delay Count Register | | Go |
| 60h | TBPHS | Time Base Phase Register | | Go |
| 62h | TBPRDHR | Time Base Period High Resolution Register | | Go |
| 63h | TBPRD | Time Base Period Register | | Go |
| 6Ah | CMPA | Counter Compare A Register | | Go |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-20. EPWM_REGS Registers (continued)**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 6Ch | CMPB | Compare B Register | | Go |
| 6Fh | CMPC | Counter Compare C Register | | Go |
| 71h | CMPD | Counter Compare D Register | | Go |
| 74h | GLDCTL2 | Global PWM Load Control Register 2 | EALLOW | Go |
| 77h | SWVDELVAL | Software Valley Mode Delay Register | | Go |
| 80h | TZSEL | Trip Zone Select Register | EALLOW | Go |
| 82h | TZDCSEL | Trip Zone Digital Comparator Select Register | EALLOW | Go |
| 84h | TZCTL | Trip Zone Control Register | EALLOW | Go |
| 85h | TZCTL2 | Additional Trip Zone Control Register | EALLOW | Go |
| 86h | TZCTLDCA | Trip Zone Control Register Digital Compare A | EALLOW | Go |
| 87h | TZCTLDCB | Trip Zone Control Register Digital Compare B | EALLOW | Go |
| 8Dh | TZEINT | Trip Zone Enable Interrupt Register | EALLOW | Go |
| 93h | TZFLG | Trip Zone Flag Register | | Go |
| 94h | TZCBCFLG | Trip Zone CBC Flag Register | | Go |
| 95h | TZOSTFLG | Trip Zone OST Flag Register | | Go |
| 97h | TZCLR | Trip Zone Clear Register | EALLOW | Go |
| 98h | TZCBCCLR | Trip Zone CBC Clear Register | EALLOW | Go |
| 99h | TZOSTCLR | Trip Zone OST Clear Register | EALLOW | Go |
| 9Bh | TZFRC | Trip Zone Force Register | EALLOW | Go |
| A4h | ETSEL | Event Trigger Selection Register | | Go |
| A6h | ETPS | Event Trigger Pre-Scale Register | | Go |
| A8h | ETFLG | Event Trigger Flag Register | | Go |
| AAh | ETCLR | Event Trigger Clear Register | | Go |
| ACh | ETFRC | Event Trigger Force Register | | Go |
| AEh | ETINTPS | Event-Trigger Interrupt Pre-Scale Register | | Go |
| B0h | ETSOCPS | Event-Trigger SOC Pre-Scale Register | | Go |
| B2h | ETCNTINITCTL | Event-Trigger Counter Initialization Control Register | | Go |
| B4h | ETCNTINIT | Event-Trigger Counter Initialization Register | | Go |
| C0h | DCTRIPSEL | Digital Compare Trip Select Register | EALLOW | Go |
| C3h | DCACTL | Digital Compare A Control Register | EALLOW | Go |
| C4h | DCBCTL | Digital Compare B Control Register | EALLOW | Go |
| C7h | DCFCTL | Digital Compare Filter Control Register | EALLOW | Go |
| C8h | DCCAPCTL | Digital Compare Capture Control Register | EALLOW | Go |
| C9h | DCFOFFSET | Digital Compare Filter Offset Register | | Go |
| CAh | DCFOFFSETCNT | Digital Compare Filter Offset Counter Register | | Go |
| CBh | DCFWINDOW | Digital Compare Filter Window Register | | Go |
| CCh | DCFWINDOWCNT | Digital Compare Filter Window Counter Register | | Go |
| CFh | DCCAP | Digital Compare Counter Capture Register | | Go |
| D2h | DCAHTRIPSEL | Digital Compare AH Trip Select | EALLOW | Go |
| D3h | DCALTRIPSEL | Digital Compare AL Trip Select | EALLOW | Go |
| D4h | DCBHTRIPSEL | Digital Compare BH Trip Select | EALLOW | Go |
| D5h | DCBLTRIPSEL | Digital Compare BL Trip Select | EALLOW | Go |
| FDh | HWVDELVAL | Hardware Valley Mode Delay Register | | Go |
| FEh | VCNTVAL | Hardware Valley Counter Register | | Go |



Copyright © 2024 Texas Instruments Incorporated

Complex bit access types are encoded to fit into small table cells. Table 15-21 shows the codes that are used for access types in this section.

**Table 15-21. EPWM_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| R-0 | R<br>-0 | Read<br>Returns 0s |
| Write Type | | |
| W | W | Write |
| W1C | W<br>1C | Write<br>1 to clear |
| W1S | W<br>1S | Write<br>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. |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.1 TBCTL Register (Offset = 0h) [Reset = 0083h]

TBCTL is shown in Figure 15-94 and described in Table 15-22.

Return to the Summary Table.

Time Base Control Register

**Figure 15-94. TBCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| FREE_SOFT | | PHSDIR | CLKDIV | | | HSPCLKDIV | |
| R/W-0h | | R/W-0h | R/W-0h | | | R/W-1h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HSPCLKDIV | SWFSYNC | SYNCOSEL | | PRDLD | PHSEN | CTRMODE | |
| R/W-1h | R-0/W1S-0h | R/W-0h | | R/W-0h | R/W-0h | R/W-3h | |

**Table 15-22. 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<br>00: Stop after the next time-base counter increment or decrement<br>01: Stop when counter completes a whole cycle:<br>- Up-count mode: stop when the time-base counter = period (TBCTR = TBPRD)<br>- Down-count mode: stop when the time-base counter = 0x00 (TBCTR = 0x00)<br>- Up-down-count mode: stop when the time-base counter = 0x00 (TBCTR = 0x00)<br>1x: Free run<br>Reset type: SYSRSn |
| 13 | PHSDIR | R/W | 0h | Phase Direction Bit<br>This bit is only used when the time-base counter is configured in the up-down-count mode. The<br>PHSDIR bit indicates the direction the time-base counter (TBCTR) will count after a synchronization<br>event occurs and a new phase value is loaded from the phase (TBPHS) register. This is<br>irrespective of the direction of the counter before the synchronization event..<br>In the up-count and down-count modes this bit is ignored.<br>0: Count down after the synchronization event.<br>1: Count up after the synchronization event.<br>Reset type: SYSRSn |
| 12-10 | CLKDIV | R/W | 0h | Time Base Clock Pre-Scale Bits<br>These bits select the time base clock pre-scale value (TBCLK = EPWMCLK/(HSPCLKDIV * CLKDIV):<br>000: /1 (default on reset)<br>001: /2<br>010: /4<br>011: /8<br>100: /16<br>101: /32<br>110: /64<br>111: /128<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-22. TBCTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 9-7 | HSPCLKDIV | R/W | 1h | High Speed Time Base Clock Pre-Scale Bits<br>These bits determine part of the time-base clock prescale value. TBCLK = EPWMCLK / (HSPCLKDIV x CLKDIV). This divisor emulates the HSPCLK in the TMS320x281x system as used on the Event Manager (EV) peripheral.<br>000: /1<br>001: /2 (default on reset)<br>010: /4<br>011: /6<br>100: /8<br>101: /10<br>110: /12<br>111: /14<br>Reset type: SYSRSn |
| 6 | SWFSYNC | R-0/W1S | 0h | Software Forced Sync Pulse<br>0: Writing a 0 has no effect and reads always return a 0.<br>1: Writing a 1 forces a one-time synchronization pulse to be generated.<br>SWFSYNC affects EPWMxSYNCO only when SYNCOSEL = 00.<br>Reset type: SYSRSn |
| 5-4 | SYNCOSEL | R/W | 0h | Sync Output Select<br>00: EPWMxSYNCI / SWFSYNC<br>01: CTR = zero: Time-base counter equal to zero (TBCTR = 0x00)<br>10: CTR = CMPB : Time-base counter equal to counter-compare B (TBCTR = CMPB)<br>11: EPWMXSYNCO is defined by TBCTL2[SYNCOSELX]<br>Reset type: SYSRSn |
| 3 | PRDLD | R/W | 0h | Active Period Reg Load from Shadow Select<br>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.<br>A write/read to the TBPRD register accesses the shadow register.<br>1: Immediate Mode (Shadow register bypassed): A write or read to the TBPRD register accesses the active register.<br>Reset type: SYSRSn |
| 2 | PHSEN | R/W | 0h | Counter Reg Load from Phase Reg Enable<br>0: Do not load the time-base counter (TBCTR) from the time-base phase register (TBPHS).<br>1: Allow Counter to be loaded from the Phase register (TBPHS) and shadow to active load events when an EPWMxSYNCI input signal occurs or a software-forced sync signal, see bit 6.<br>Reset type: SYSRSn |
| 1-0 | CTRMODE | R/W | 3h | Counter Mode<br>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:<br>00: Up-count mode<br>01: Down-count mode<br>10: Up-down count mode<br>11: Freeze counter operation (default on reset)<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.2 TBCTL2 Register (Offset = 1h) [Reset = 0000h]

TBCTL2 is shown in Figure 15-95 and described in Table 15-23.

Return to the Summary Table.

Time Base Control Register 2

**Figure 15-95. TBCTL2 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| PRDLDSYNC | | SYNCOSELX | | RESERVED | | | |
| R/W-0h | | R/W-0h | | R-0-0h | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OSHTSYNC | OSHTSYNCMODE | RESERVED | RESERVED | | | | |
| R-0/W1S-0h | R/W-0h | R/W-0h | R-0-0h | | | | |

**Table 15-23. TBCTL2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-14 | PRDLDSYNC | R/W | 0h | Shadow to Active Period Register Load on SYNC event<br>00: Shadow to Active Load of TBPRD occurs only when TBCTR = 0 (same as legacy).<br>01: Shadow to Active Load of TBPRD occurs both when TBCTR = 0 and when SYNC occurs.<br>10: Shadow to Active Load of TBPRD occurs only when a SYNC is received.<br>11: Reserved<br>Note: This bit selection is valid only if TBCTL[PRDLD]=0.<br>Reset type: SYSRSn |
| 13-12 | SYNCOSELX | R/W | 0h | Extended selection bits for SYNCOUT<br>00: Disabled EPWMxSYNCO sync signal<br>01: EPWMxSYNCO = CMPC<br>10: EPWMxSYNCO = CMPD<br>11: Reserved<br>Reset type: SYSRSn |
| 11-8 | RESERVED | R-0 | 0h | Reserved |
| 7 | OSHTSYNC | R-0/W1S | 0h | Oneshot sync bit<br>0: Writing a '0' has no effect.<br>1: Allow one sync pulse to propogate.<br>Reset type: SYSRSn |
| 6 | OSHTSYNCMODE | R/W | 0h | Oneshot sync enable bit<br>0: Oneshot sync mode disabled<br>1: Oneshot sync mode enabled<br>Reset type: SYSRSn |
| 5 | RESERVED | R/W | 0h | Reserved |
| 4-0 | RESERVED | R-0 | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.3 TBCTR Register (Offset = 4h) [Reset = 0000h]

TBCTR is shown in Figure 15-96 and described in Table 15-24.

Return to the Summary Table.

Time Base Counter Register

**Figure 15-96. 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 15-24. TBCTR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | TBCTR | R/W | 0h | Time Base Counter Register<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.4 TBSTS Register (Offset = 5h) [Reset = 0001h]

TBSTS is shown in Figure 15-97 and described in Table 15-25.

Return to the Summary Table.

Time Base Status Register

**Figure 15-97. 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 | CTRDIR |
| R-0-0h | | | | | R/W1C-0h | R/W1C-0h | R-1h |

**Table 15-25. 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<br>0: Reading a 0 indicates the time-base counter never reached its maximum value. Writing a 0 will have no effect.<br>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.<br>Reset type: SYSRSn |
| 1 | SYNCI | R/W1C | 0h | Input Synchronization Latched Status Bit<br>0: Writing a 0 will have no effect. Reading a 0 indicates no external synchronization event has occurred.<br>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.<br>Reset type: SYSRSn |
| 0 | CTRDIR | R | 1h | Time Base Counter Direction Status Bit<br>0: Time-Base Counter is currently counting down.<br>1: Time-Base Counter is currently counting up.<br>Note: This bit is only valid when the counter is not frozen.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.5 CMPCTL Register (Offset = 8h) [Reset = 0000h]

CMPCTL is shown in Figure 15-98 and described in Table 15-26.

Return to the Summary Table.

Counter Compare Control Register

**Figure 15-98. CMPCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | LOADBSYNC | | LOADASYNC | | SHDWBFULL | SHDWAFULL |
| R-0-0h | | R/W-0h | | R/W-0h | | R-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | SHDWBMODE | RESERVED | SHDWAMODE | LOADBMODE | | LOADAMODE | |
| R-0-0h | R/W-0h | R-0-0h | R/W-0h | R/W-0h | | R/W-0h | |

**Table 15-26. 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<br>00: Shadow to Active Load of CMPB:CMPBHR occurs according to LOADBMODE (bits 1,0) (same as legacy)<br>01: Shadow to Active Load of CMPB:CMPBHR occurs both according to LOADBMODE bits and when SYNC occurs<br>10: Shadow to Active Load of CMPB:CMPBHR occurs only when a SYNC is received<br>11: Reserved<br>Note: This bit is valid only if CMPCTL[SHDWBMODE] = 0.<br>Reset type: SYSRSn |
| 11-10 | LOADASYNC | R/W | 0h | Shadow to Active CMPA Register Load on SYNC event<br>00: Shadow to Active Load of CMPA:CMPAHR occurs according to LOADAMODE (bits 1,0) (same as legacy)<br>01: Shadow to Active Load of CMPA:CMPAHR occurs both according to LOADAMODE bits and when SYNC occurs<br>10: Shadow to Active Load of CMPA:CMPAHR occurs only when a SYNC is received<br>11: Reserved<br>Note: This bit is valid only if CMPCTL[SHDWAMODE] = 0.<br>Reset type: SYSRSn |
| 9 | SHDWBFULL | R | 0h | Counter-compare B (CMPB) Shadow Register Full Status Flag<br>This bit self clears once a loadstrobe occurs.<br>0: CMPB shadow register not full yet<br>1: Indicates the CMPB shadow register is full<br>a CPU write will overwrite current shadow value<br>Reset type: SYSRSn |
| 8 | SHDWAFULL | R | 0h | Counter-compare A (CMPA) Shadow Register Full Status Flag<br>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.<br>0: CMPA shadow register not full yet<br>1: Indicates the CMPA shadow register is full, a CPU write will overwrite the current shadow value<br>Reset type: SYSRSn |
| 7 | RESERVED | R-0 | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-26. CMPCTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 6 | SHDWBMODE | R/W | 0h | Counter-compare B (CMPB) Register Operating Mode<br>0: Shadow mode. Operates as a double buffer. All writes via the CPU access the shadow register<br>1: Immediate mode. Only the active compare B register is used. All writes and reads directly access the active register for immediate compare action<br>Reset type: SYSRSn |
| 5 | RESERVED | R-0 | 0h | Reserved |
| 4 | SHDWAMODE | R/W | 0h | Counter-compare A (CMPA) Register Operating Mode<br>0: Shadow mode. Operates as a double buffer. All writes via the CPU access the shadow register<br>1: Immediate mode. Only the active compare register is used. All writes and reads directly access the active register for immediate compare action<br>Reset type: SYSRSn |
| 3-2 | LOADBMODE | R/W | 0h | Active Counter-Compare B (CMPB) Load From Shadow Select Mode<br>This bit has no effect in immediate mode (CMPCTL[SHDWBMODE] = 1).<br>00: Load on CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000)<br>01: Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD)<br>10: Load on either CTR = Zero or CTR = PRD<br>11: Freeze (no loads possible)<br>Reset type: SYSRSn |
| 1-0 | LOADAMODE | R/W | 0h | Active Counter-Compare A (CMPA) Load From Shadow Select Mode<br>This bit has no effect in immediate mode (CMPCTL[SHDWAMODE] = 1).<br>00: Load on CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000)<br>01: Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD)<br>10: Load on either CTR = Zero or CTR = PRD<br>11: Freeze (no loads possible)<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.6 CMPCTL2 Register (Offset = 9h) [Reset = 0000h]

CMPCTL2 is shown in Figure 15-99 and described in Table 15-27.

Return to the Summary Table.

Counter Compare Control Register 2

**Figure 15-99. CMPCTL2 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | LOADDSYNC | | LOADCSYNC | | RESERVED | |
| R-0-0h | | R/W-0h | | R/W-0h | | R-0-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | SHDWDMODE | RESERVED | SHDWCMODE | LOADDMODE | | LOADCMODE | |
| R-0-0h | R/W-0h | R-0-0h | R/W-0h | R/W-0h | | R/W-0h | |

**Table 15-27. 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<br>00: Shadow to Active Load of CMPD occurs according to LOADDMODE<br>01: Shadow to Active Load of CMPD occurs both according to LOADDMODE bits and when SYNC occurs<br>10: Shadow to Active Load of CMPD occurs only when a SYNC is received<br>11: Reserved<br>Note: This bit is valid only if CMPCTL2[SHDWDMODE] = 0.<br>Reset type: SYSRSn |
| 11-10 | LOADCSYNC | R/W | 0h | Shadow to Active CMPC Register Load on SYNC event<br>00: Shadow to Active Load of CMPC occurs according to LOADCMODE<br>01: Shadow to Active Load of CMPC occurs both according to LOADCMODE bits and when SYNC occurs<br>10: Shadow to Active Load of CMPC occurs only when a SYNC is received<br>11: Reserved<br>Note: This bit is valid only if CMPCTL2[SHDWCMODE] = 0.<br>Reset type: SYSRSn |
| 9-7 | RESERVED | R-0 | 0h | Reserved |
| 6 | SHDWDMODE | R/W | 0h | Counter-Compare D Register Operating Mode<br>0: Shadow mode - operates as a double buffer. All writes via the CPU access Shadow register.<br>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.<br>Reset type: SYSRSn |
| 5 | RESERVED | R-0 | 0h | Reserved |
| 4 | SHDWCMODE | R/W | 0h | Counter-Compare C Register Operating Mode<br>0: Shadow mode - operates as a double buffer. All writes via the CPU access Shadow register.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-27. 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<br>00: Load on CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000)<br>01: Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD)<br>10: Load on either CTR = Zero or CTR = PRD<br>11: Freeze (no loads possible)<br>Note: Has no effect in Immediate mode.<br>Reset type: SYSRSn |
| 1-0 | LOADCMODE | R/W | 0h | Active Counter-Compare C (CMPC) Load from Shadow Select Mode<br>00: Load on CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000)<br>01: Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD)<br>10: Load on either CTR = Zero or CTR = PRD<br>11: Freeze (no loads possible)<br>Note: Has no effect in Immediate mode.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.7 DBCTL Register (Offset = Ch) [Reset = 0000h]

DBCTL is shown in Figure 15-100 and described in Table 15-28.

Return to the Summary Table.

Dead-Band Generator Control Register

**Figure 15-100. DBCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| HALFCYCLE | DEDB_MODE | OUTSWAP | | SHDWDBFED MODE | SHDWDBRED MODE | LOADFEDMODE | |
| R/W-0h | R/W-0h | R/W-0h | | R/W-0h | R/W-0h | R/W-0h | |
| 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 15-28. DBCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | HALFCYCLE | R/W | 0h | Half Cycle Clocking Enable Bit<br>0: Full cycle clocking enabled. The dead-band counters are clocked at the TBCLK rate.<br>1: Half cycle clocking enabled. The dead-band counters are clocked at TBCLK*2.<br>Reset type: SYSRSn |
| 14 | DEDB_MODE | R/W | 0h | Dead Band Dual-Edge B Mode Control (S8 switch)<br>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.<br>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<br>otherwise, OutA will be invalid.<br>Reset type: SYSRSn |
| 13-12 | OUTSWAP | R/W | 0h | Dead Band Output Swap Control<br>Bit 13 controls the S6 switch and bit 12 controls the S7 switch.<br>00: OutA and OutB signals are as defined by OUT-MODE bits.<br>01: OutA = A-path as defined by OUT-MODE bits.<br>OutB = A-path as defined by OUT-MODE bits (rising edge delay or delay-bypassed A signal path).<br>10: OutA = B-path as defined by OUT-MODE bits (falling edge delay or delay-bypassed B signal path).<br>OutB = B-path as defined by OUT-MODE bits.<br>11: OutA = B-path as defined by OUT-MODE bits (falling edge delay or delay-bypassed B signal path).<br>OutB = A-path as defined by OUT-MODE bits (rising edge delay or delay-bypassed A signal path).<br>Reset type: SYSRSn |
| 11 | SHDWDBFEDMODE | R/W | 0h | FED Dead-Band Load Mode<br>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.'<br>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).<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-28. DBCTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 10 | SHDWDBREDMODE | R/W | 0h | RED Dead-Band Load Mode<br>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.'<br>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).<br>Reset type: SYSRSn |
| 9-8 | LOADFEDMODE | R/W | 0h | Active DBFED Load from Shadow Select Mode<br>00: Load on Counter = 0 (CNT_eq)<br>01: Load on Counter = Period (PRD_eq)<br>10: Load on either Counter = 0, or Counter = Period<br>11: Freeze (no loads possible)<br>Note: has no effect in Immediate mode.<br>Reset type: SYSRSn |
| 7-6 | LOADREDMODE | R/W | 0h | Active DBRED Load from Shadow Select Mode<br>00: Load on Counter = 0 (CNT_eq)<br>01: Load on Counter = Period (PRD_eq)<br>10: Load on either Counter = 0, or Counter = Period<br>11: Freeze (no loads possible)<br>Note: has no effect in Immediate mode.<br>Reset type: SYSRSn |
| 5-4 | IN_MODE | R/W | 0h | Dead-Band Input Mode Control<br>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.<br>00: EPWMxA In (from the action-qualifier) is the source for both falling-edge and rising-edge delay.<br>01: EPWMxB In (from the action-qualifier) is the source for rising-edge delayed signal.<br>EPWMxA In (from the action-qualifier) is the source for falling-edge delayed signal.<br>10: EPWMxA In (from the action-qualifier) is the source for rising-edge delayed signal.<br>EPWMxB In (from the action-qualifier) is the source for falling-edge delayed signal.<br>11: EPWMxB In (from the action-qualifier) is the source for both rising-edge delay and falling-edge delayed signal.<br>Reset type: SYSRSn |
| 3-2 | POLSEL | R/W | 0h | Polarity Select Control<br>Bit 3 controls the S3 switch and bit 2 controls the S2 switch.<br>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.<br>00: Active high (AH) mode. Neither EPWMxA nor EPWMxB is inverted (default).<br>01: Active low complementary (ALC) mode. EPWMxA is inverted.<br>10: Active high complementary (AHC). EPWMxB is inverted.<br>11: Active low (AL) mode. Both EPWMxA and EPWMxB are inverted.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-28. DBCTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1-0 | OUT_MODE | R/W | 0h | Dead-Band Output Mode Control<br>Bit 1 controls the S1 switch and bit 0 controls the S0 switch.<br>00: DBM is fully disabled or by-passed. In this mode the POLSEL and IN-MODE bits have no effect.<br>01: Apath = InA (delay is by-passed for A signal path)<br>Bpath = FED (Falling Edge Delay in B signal path)<br>10: Apath = RED (Rising Edge Delay in A signal path)<br>Bpath = InB (delay is by-passed for B signal path)<br>11: DBM is fully enabled (i.e. both RED and FED active)<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.8 DBCTL2 Register (Offset = Dh) [Reset = 0000h]

DBCTL2 is shown in Figure 15-101 and described in Table 15-29.

Return to the Summary Table.

Dead-Band Generator Control Register 2

**Figure 15-101. DBCTL2 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | SHDWDBCTLMODE | LOADDBCTLMODE | |
| R-0-0h | | | | | R/W-0h | R/W-0h | |

**Table 15-29. DBCTL2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-3 | RESERVED | R-0 | 0h | Reserved |
| 2 | SHDWDBCTLMODE | R/W | 0h | DBCTL Load Mode<br>0: Immediate mode - only the Active DBCTL register is used. All writes/reads via the CPU directly access the Active register.<br>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.<br>Reset type: SYSRSn |
| 1-0 | LOADDBCTLMODE | R/W | 0h | Active DBCTL Load from Shadow Select Mode<br>00: Load on Counter = 0 (CNT_eq)<br>01: Load on Counter = Period (PRD_eq)<br>10: Load on either Counter = 0, or Counter = Period<br>11: Freeze (no loads possible)<br>Note: has no effect in Immediate mode<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.9 AQCTL Register (Offset = 10h) [Reset = 0000h]

AQCTL is shown in Figure 15-102 and described in Table 15-30.

Return to the Summary Table.

Action Qualifier Control Register

**Figure 15-102. AQCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | LDAQBSYNC | | LDAQASYNC | |
| R-0-0h | | | | R/W-0h | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | SHDWAQBMODE | RESERVED | SHDWAQAMODE | LDAQBMODE | | LDAQAMODE | |
| R-0-0h | R/W-0h | R-0-0h | R/W-0h | R/W-0h | | R/W-0h | |

**Table 15-30. 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<br>00: Shadow to Active Load of AQCTLB occurs according to LDAQBMODE<br>01: Shadow to Active Load of AQCTLB occurs both according to LDAQBMODE bits and when SYNC occurs.<br>10: Shadow to Active Load of AQCTLB occurs only when a SYNC is received.<br>11: Reserved<br>Note: This bit is valid only if AQCTL[SHDWAQBMODE] = 1.<br>Reset type: SYSRSn |
| 9-8 | LDAQASYNC | R/W | 0h | Shadow to Active AQCTLA Register Load on SYNC event<br>00: Shadow to Active Load of AQCTLA occurs according to LDAQAMODE<br>01: Shadow to Active Load of AQCTLA occurs both according to LDAQAMODE bits and when SYNC occurs.<br>10: Shadow to Active Load of AQCTLA occurs only when a SYNC is received.<br>11: Reserved<br>Note: This bit is valid only if AQCTL[SHDWAQAMODE] = 1.<br>Reset type: SYSRSn |
| 7 | RESERVED | R-0 | 0h | Reserved |
| 6 | SHDWAQBMODE | R/W | 0h | Action Qualifier B Register operating mode<br>1: Shadow mode - operates as a double buffer. All writes via the CPU access Shadow register.<br>0: Immediate mode - only the Active action qualifier register is used. All writes/reads via the CPU directly access the Active register.<br>Reset type: SYSRSn |
| 5 | RESERVED | R-0 | 0h | Reserved |
| 4 | SHDWAQAMODE | R/W | 0h | Action Qualifier A Register operating mode<br>1: Shadow mode - operates as a double buffer. All writes via the CPU access Shadow register.<br>0: Immediate mode - only the Active action qualifier register is used. All writes/reads via the CPU directly access the Active register.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-30. 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<br>00: Load on CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000)<br>01: Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD)<br>10: Load on either CTR = Zero or CTR = PRD<br>11: Freeze (no loads possible)<br>Note: has no effect in Immediate mode.<br>Reset type: SYSRSn |
| 1-0 | LDAQAMODE | R/W | 0h | Active Action Qualifier A Load from Shadow Select Mode<br>00: Load on CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000)<br>01: Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD)<br>10: Load on either CTR = Zero or CTR = PRD<br>11: Freeze (no loads possible)<br>Note: has no effect in Immediate mode.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.10 AQTSRCSEL Register (Offset = 11h) [Reset = 0000h]

AQTSRCSEL is shown in Figure 15-103 and described in Table 15-31.

Return to the Summary Table.

Action Qualifier Trigger Event Source Select Register

**Figure 15-103. AQTSRCSEL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| T2SEL | | | | T1SEL | | | |
| R/W-0h | | | | R/W-0h | | | |

**Table 15-31. 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<br>0000: DCAEVT1<br>0001: DCAEVT2<br>0010: DCBEVT1<br>0011: DCBEVT2<br>0100: TZ1<br>0101: TZ2<br>0110: TZ3<br>0111: EPWMxSYNCI<br>1xxx: Reserved<br>Reset type: SYSRSn |
| 3-0 | T1SEL | R/W | 0h | T1 Event Source Select Bits<br>0000: DCAEVT1<br>0001: DCAEVT2<br>0010: DCBEVT1<br>0011: DCBEVT2<br>0100: TZ1<br>0101: TZ2<br>0110: TZ3<br>0111: EPWMxSYNCI<br>1xxx: Reserved<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.11 PCCTL Register (Offset = 14h) [Reset = 0000h]

PCCTL is shown in Figure 15-104 and described in Table 15-32.

Return to the Summary Table.

PWM Chopper Control Register

**Figure 15-104. PCCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | CHPDUTY | |
| R-0-0h | | | | | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CHPFREQ | | | OSHTWTH | | | | CHPEN |
| R/W-0h | | | R/W-0h | | | | R/W-0h |

**Table 15-32. 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<br>000: Duty = 1/8 (12.5%)<br>001: Duty = 2/8 (25.0%)<br>010: Duty = 3/8 (37.5%)<br>011: Duty = 4/8 (50.0%)<br>100: Duty = 5/8 (62.5%)<br>101: Duty = 6/8 (75.0%)<br>110: Duty = 7/8 (87.5%)<br>111: Reserved<br>Reset type: SYSRSn |
| 7-5 | CHPFREQ | R/W | 0h | Chopping Clock Frequency<br>000: Divide by 1 (no prescale, = 12.5 MHz at 100 MHz TBCLK)<br>001: Divide by 2 (6.25 MHz at 100 MHz TBCLK)<br>010: Divide by 3 (4.16 MHz at 100 MHz TBCLK)<br>011: Divide by 4 (3.12 MHz at 100 MHz TBCLK)<br>100: Divide by 5 (2.50 MHz at 100 MHz TBCLK)<br>101: Divide by 6 (2.08 MHz at 100 MHz TBCLK)<br>110: Divide by 7 (1.78 MHz at 100 MHz TBCLK)<br>111: Divide by 8 (1.56 MHz at 100 MHz TBCLK)<br>Reset type: SYSRSn |
| 4-1 | OSHTWTH | R/W | 0h | One-Shot Pulse Width<br>0000: 1 x EPWMCLK / 8 wide ( = 80 ns at 100 MHz EPWMCLK)<br>0001: 2 x EPWMCLK / 8 wide ( = 160 ns at 100 MHz EPWMCLK)<br>0010: 3 x EPWMCLK / 8 wide ( = 240 ns at 100 MHz EPWMCLK)<br>0011: 4 x EPWMCLK / 8 wide ( = 320 ns at 100 MHz EPWMCLK)<br>0100: 5 x EPWMCLK / 8 wide ( = 400 ns at 100 MHz EPWMCLK)<br>0101: 6 x EPWMCLK / 8 wide ( = 480 ns at 100 MHz EPWMCLK)<br>0110: 7 x EPWMCLK / 8 wide ( = 560 ns at 100 MHz EPWMCLK)<br>0111: 8 x EPWMCLK / 8 wide ( = 640 ns at 100 MHz EPWMCLK)<br>1000: 9 x EPWMCLK / 8 wide ( = 720 ns at 100 MHz EPWMCLK)<br>1001: 10 x EPWMCLK / 8 wide ( = 800 ns at 100 MHz EPWMCLK)<br>1010: 11 x EPWMCLK / 8 wide ( = 880 ns at 100 MHz EPWMCLK)<br>1011: 12 x EPWMCLK / 8 wide ( = 960 ns at 100 MHz EPWMCLK)<br>1100: 13 x EPWMCLK / 8 wide ( = 1040 ns at 100 MHz EPWMCLK)<br>1101: 14 x EPWMCLK / 8 wide ( = 1120 ns at 100 MHz EPWMCLK)<br>1110: 15 x EPWMCLK / 8 wide ( = 1200 ns at 100 MHz EPWMCLK)<br>1111: 16 x EPWMCLK / 8 wide ( = 1280 ns at 100 MHz EPWMCLK)<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-32. PCCTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | CHPEN | R/W | 0h | PWM-Chopping Enable<br>0: Disable (bypass) PWM chopping function<br>1: Enable chopping function<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.12 VCAPCTL Register (Offset = 18h) [Reset = 0000h]

VCAPCTL is shown in Figure 15-105 and described in Table 15-33.

Return to the Summary Table.

Valley Capture Control Register

**Figure 15-105. VCAPCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | EDGEFILTDLYSEL | 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 15-33. 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<br>0: No delay applied to the edge filter output<br>1: HWDELAYVAL delay applied to the edge filter output<br>Reset type: SYSRSn |
| 9-7 | VDELAYDIV | R/W | 0h | Valley Delay Mode Divide Enable<br>000: HWVDELVAL = SWVDELVAL<br>001: HWVDELVAL = VCNTVAL+SWVDELVAL<br>010: HWVDELVAL = VCNTVAL>>1+SWVDELVAL<br>011: HWVDELVAL = VCNTVAL>>2+SWVDELVAL<br>100: HWVDELVAL = VCNTVAL>>4+SWVDELVAL<br>Note: Delay value between the consecutive edge captures can optionally be divided by using these bits.<br>Reset type: SYSRSn |
| 6-5 | RESERVED | R-0 | 0h | Reserved |
| 4-2 | TRIGSEL | R/W | 0h | Status of Numbered of Captured Events<br>000: Capture sequence is triggered by software via writes to VCAPCTL[VCAPSTART].<br>001: Capture sequence is triggered by CNT_zero event.<br>010: Capture sequence is triggered by PRD_eq event.<br>011: Capture sequence is triggered by CNT_zero or PRD_eq event.<br>100: Capture sequence is triggered by DCAEVT1 event.<br>101: Capture sequence is triggered by DCAEVT2 event.<br>110: Capture sequence is triggered by DCBEVT1 event.<br>111: Capture sequence is triggered by DCBEVT2 event.<br>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[SRCSEL] register.<br>Note: Same event may not be chosen in both DCFCTL[SRCSEL] and VCAPCTL[TRIGSEL] registers.<br>Note: Once the chosen event in VCAPCTL[TRIGSEL] occurs, irrespective of the current capture status, capture sequence is retriggered.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-33. VCAPCTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1 | VCAPSTART | R-0/W1S | 0h | Valley Capture Start<br>0: Writing a 0 has no effect<br>1: Trigger the capture sequence once if VCAPCTL[TRIGSEL]=0x0<br>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.<br>Reset type: SYSRSn |
| 0 | VCAPE | R/W | 0h | Valley Capture Enable/Disable<br>0: Disabled<br>1: Enabled<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.13 VCNTCFG Register (Offset = 19h) [Reset = 0000h]

VCNTCFG is shown in Figure 15-106 and described in Table 15-34.

Return to the Summary Table.

Valley Counter Config Register

**Figure 15-106. 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 15-34. VCNTCFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | STOPEDGESTS | R | 0h | Stop Edge Status Bit<br>0: Stop edge has not occurred<br>1: Stop edge occurred<br>Note: This bit is set only after the trigger sequence is armed (upon occurrence of trigger pulse selected through VCAPCTL[TRIGSEL]) and STOPEDGE occurs.<br>Note:This bit is reset by the occurrence of the trigger pulse selected through VCAPCTL[TRIGSEL]<br>Reset type: SYSRSn |
| 14-12 | RESERVED | R-0 | 0h | Reserved |
| 11-8 | STOPEDGE | R/W | 0h | Counter Stop Edge Selection<br>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:<br>0000: Do not stop<br>0001<br>1st edge<br>0010: 2nd edge<br>0011: 3rd edge<br>...<br>1,1,1,1: 15th edge<br>Reset type: SYSRSn |
| 7 | STARTEDGESTS | R | 0h | Start Edge Status Bit<br>0: Start edge has not occurred<br>1: Start edge occurred<br>Note: This bit is set only after the trigger sequence is armed (upon occurrence of trigger pulse selected through VCAPCTL[TRIGSEL]) and STARTEDGE occurs.<br>Note:This bit is reset by the occurrence of the trigger pulse selected through VCAPCTL[TRIGSEL]<br>Reset type: SYSRSn |
| 6-4 | RESERVED | R-0 | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-34. VCNTCFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-0 | STARTEDGE | R/W | 0h | Counter Start Edge Selection<br>Once the counter operation is armed, upon occurrence of trigger pulse selected through VCAPCTL[TRIGSEL] pulse - valley counter would start counting upon the occurrence of chosen number of events thorough this bit field. Start counting on occurrence of<br>0000: Do not start<br>0001: 1st edge<br>0010: 2nd edge<br>0011: 3rd edge<br>...<br>1111: 15th edge<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.14 HRCNFG Register (Offset = 20h) [Reset = 0000h]

HRCNFG is shown in Figure 15-107 and described in Table 15-35.

Return to the Summary Table.

HRPWM Configuration Register
This register is only accessible on EPWM modules with HRPWM capabilities.

**Figure 15-107. 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 15-35. 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<br>Selects the time event that loads the CMPBHR shadow value into the active register.<br>00: Load on CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000)<br>01: Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD)<br>10: Load on either CTR = Zero or CTR = PRD<br>11: Reserved<br>Reset type: SYSRSn |
| 10 | CTLMODEB | R/W | 0h | Control Mode Bits<br>Selects the register (CMP/TBPRD or TBPHS) that controls the MEP:<br>0: CMPBHR(8) or TBPRDHR(8) Register controls the edge position (i.e., this is duty or period control mode). (Default on Reset)<br>1: TBPHSHR(8) Register controls the edge position (i.e., this is phase control mode).<br>Reset type: SYSRSn |
| 9-8 | EDGMODEB | R/W | 0h | Edge Mode Bits<br>Selects the edge of the PWM that is controlled by the micro-edge position (MEP) logic:<br>00: HRPWM capability is disabled (default on reset)<br>01: MEP control of rising edge (CMPBHR)<br>10: MEP control of falling edge (CMPBHR)<br>11: MEP control of both edges (TBPHSHR or TBPRDHR)<br>Reset type: SYSRSn |
| 7 | SWAPAB | R/W | 0h | Swap ePWM A & B Output Signals<br>This bit enables the swapping of the A & B signal outputs. The selection is as follows:<br>0: ePWMxA and ePWMxB outputs are unchanged.<br>1: ePWMxA signal appears on ePWMxB output and ePWMxB signal appears on ePWMxA output.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-35. HRCNFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 6 | AUTOCONV | R/W | 0h | Auto Convert Delay Line Value<br>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.<br>0: Automatic HRMSTEP scaling is disabled.<br>1: Automatic HRMSTEP scaling is enabled.<br>If application software is manually scaling the fractional duty cycle, or phase (i.e. software sets CMPAHR = (fraction(PWMduty * PWMperiod) * MEP Scale Factor)<<8 + 0x080 for duty cycle), then this mode must be disabled.<br>Reset type: SYSRSn |
| 5 | SELOUTB | R/W | 0h | EPWMxB Output Select Bit<br>This bit selects which signal is output on the ePWMxB channel output.<br>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.<br>0: ePWMxB output is normal.<br>1: ePWMxB output is inverted version of ePWMxA signal.<br>Reset type: SYSRSn |
| 4-3 | HRLOAD | R/W | 0h | Shadow Mode Bit<br>Selects the time event that loads the CMPAHR shadow value into the active register.<br>00: Load on CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000)<br>01: Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD)<br>10: Load on either CTR = Zero or CTR = PRD<br>11: Reserved<br>Reset type: SYSRSn |
| 2 | CTLMODE | R/W | 0h | Control Mode Bits<br>Selects the register (CMP/TBPRD or TBPHS) that controls the MEP:<br>0: CMPAHR(8) or TBPRDHR(8) Register controls the edge position (i.e., this is duty or period control mode). (Default on Reset)<br>1: TBPHSHR(8) Register controls the edge position (i.e., this is phase control mode).<br>Reset type: SYSRSn |
| 1-0 | EDGMODE | R/W | 0h | Edge Mode Bits<br>Selects the edge of the PWM that is controlled by the micro-edge position (MEP) logic:<br>00: HRPWM capability is disabled (default on reset)<br>01: MEP control of rising edge (CMPAHR)<br>10: MEP control of falling edge (CMPAHR)<br>11: MEP control of both edges (TBPHSHR or TBPRDHR)<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.15 HRPWR Register (Offset = 21h) [Reset = 0000h]

HRPWR is shown in Figure 15-108 and described in Table 15-36.

Return to the Summary Table.

HRPWM Power Register
This register is only accessible on EPWM modules with HRPWM capabilities.

**Figure 15-108. 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 15-36. HRPWR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | CALPWRON | R/W | 0h | MEP Calibration Power Bits (only available on ePWM1)<br>0: Disables MEP calibration logic in the HRPWM and reduces power consumption.<br>1: Enables MEP calibration logic<br>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 |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.16 HRMSTEP Register (Offset = 26h) [Reset = 0000h]

HRMSTEP is shown in Figure 15-109 and described in Table 15-37.

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 15-109. HRMSTEP Register**

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

**Table 15-37. 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<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.17 HRCNFG2 Register (Offset = 27h) [Reset = 0000h]

HRCNFG2 is shown in Figure 15-110 and described in Table 15-38.

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 15-110. 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 15-38. 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]<br>Selects the time event that loads the DBFEDHR shadow value into the active register.<br>00 Load on CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000)<br>01 Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD)<br>10 Load on either CTR = Zero or CTR = PRD<br>11 Reserved<br>Reset type: SYSRSn |
| 3-2 | CTLMODEDBRED | R/W | 0h | Shadow Mode Bit - selection should match DBCTL[LOADREDMODE]<br>Selects the time event that loads the DBREDHR shadow value into the active register.<br>00 Load on CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000)<br>01 Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD)<br>10 Load on either CTR = Zero or CTR = PRD<br>11 Reserved<br>Reset type: SYSRSn |
| 1-0 | EDGMODEDB | R/W | 0h | Edge Mode Bits<br>Selects the edge of the PWM that is controlled by the micro-edge position (MEP) logic:<br>00 HRPWM capability is disabled (default on reset)<br>01 MEP control of rising edge (DBREDHR)<br>10 MEP control of falling edge (DBFEDHR)<br>11 MEP control of both edges (rising edge of DBREDHR or falling edge of DBFEDHR )<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.18 HRPCTL Register (Offset = 2Dh) [Reset = 0000h]

HRPCTL is shown in Figure 15-111 and described in Table 15-39.

Return to the Summary Table.

High Resolution Period Control Register
This register is only accessible on EPWM modules with HRPWM capabilities.

**Figure 15-111. HRPCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | PWMSYNCSELX | | | RESERVED | TBPHSHRLOADE | PWMSYNCSEL | HRPE |
| R-0-0h | R/W-0h | | | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 15-39. 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<br>000: EPWMSYNCPER is defined by PWMSYNCSEL - > default condition (compatible with previous EPWM versions)<br>001: Reserved<br>010: Reserved<br>011: Reserved<br>100: CTR = CMPC, Count direction Up<br>101: CTR = CMPC, Count direction Down<br>110: CTR = CMPD, Count direction Up<br>111: CTR = CMPD, Count direction Down<br>Reset type: SYSRSn |
| 3 | RESERVED | R/W | 0h | Reserved |
| 2 | TBPHSHRLOADE | R/W | 0h | TBPHSHR Load Enable<br>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.<br>0: Disables synchronization of high-resolution phase on a SYNCIN, TBCTL[SWFSYNC] or digital compare event:<br>1: Synchronize the high-resolution phase on a SYNCIN, TBCTL[SWFSYNC] or digital comparator synchronization event. The phase is synchronized using the contents of the high-resolution phase TBPHSHR register. The TBCTL[PHSEN] bit which enables the loading of the TBCTR register with TBPHS register value on a SYNCIN or TBCTL[SWFSYNC] event works independently. However, users need to enable this bit also if they want to control phase in conjunction with the high-resolution period feature.<br>This bit and the TBCTL[PHSEN] bit must be set to 1 when high-resolution period is enabled for up-down count mode even if TBPHSHR = 0x0000. This bit does not need to be set when only high-resolution duty is enabled.<br>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:<br>0 CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD)<br>1 CTR = zero: Time-base counter equal to zero (TBCTR = 0x00)<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-39. HRPCTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | HRPE | R/W | 0h | High Resolution Period Enable Bit<br>0: High resolution period feature disabled. In this mode the ePWM behaves as a Type 4 ePWM.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.19 TRREM Register (Offset = 2Eh) [Reset = 0000h]

TRREM is shown in Figure 15-112 and described in Table 15-40.

Return to the Summary Table.

HRPWM High Resolution Remainder Register
This register is only accessible on EPWM modules with HRPWM capabilities.

**Figure 15-112. TRREM Register**

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

**Table 15-40. TRREM Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-11 | RESERVED | R-0 | 0h | Reserved |
| 10-0 | TRREM | R/W | 0h | HRPWM Remainder Bits: This 11-bit value keeps track of the remainder portion of the HRPWM algorithm calculations.<br>This value keeps track of the remainder portion of the HRPWM hardware calculations.<br>Notes:<br>1. The lower 8-bits of the TRREM register can be automatically initialized with the TBPHSHR value on a SYNCIN or TBCTL[SWFSYNC] event or DC event (if enabled). The user can also write a value with the CPU.<br>2. Priority of TRREM register updates:<br>Sync (software or hardware) TBPHSHR copied to TRREM : Highest Priority<br>HRPWM Hardware (updates TRREM register): Next priority<br>CPU Write To TRREM Register: Lowest Priority<br>3. Bit 10 of TRREM register is not used in asymmetrical mode. This bit can be forced to zero.<br>TRREM will be initialized to 0x0 and 0x100 in Up and Up-down modes respectively.<br>Asymmetrical Mode:<br>TRREM[7:0] = TBPHSHR[15:8]<br>TRREM[10,9,8] = 0,0,0<br>Symmetrical Mode:<br>TRREM[7:0] = TBPHSHR[15:8]<br>TRREM[10,9,8] = 0,0,1<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.20 GLDCTL Register (Offset = 34h) [Reset = 0000h]

GLDCTL is shown in Figure 15-113 and described in Table 15-41.

Return to the Summary Table.

Global PWM Load Control Register

**Figure 15-113. GLDCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | GLDCNT | | | GLDPRD | |
| R-0-0h | | | R-0h | | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GLDPRD | RESERVED | OSHTMODE | GLDMODE | | | | GLD |
| R/W-0h | R-0-0h | R/W-0h | R/W-0h | | | | R/W-0h |

**Table 15-41. 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<br>These bits indicate how many selected events have occurred:<br>000: No events<br>001: 1 event<br>010: 2 events<br>011: 3 events<br>100: 4 events<br>101: 5 events<br>110: 6 events<br>111: 7 events<br>Reset type: SYSRSn |
| 9-7 | GLDPRD | R/W | 0h | Global Load Strobe Period Select Register<br>These bits select how many selected events need to occur before a load strobe is generated<br>000: Disable counter<br>001: Generate strobe on GLDCNT = 001 (1st event)<br>010: Generate strobe on GLDCNT = 010 (2nd event)<br>011: Generate strobe on GLDCNT = 011 (3rd event)<br>100: Generate strobe on GLDCNT = 011 (4th event)<br>101: Generate strobe on GLDCNT = 001 (5th event)<br>110: Generate strobe on GLDCNT = 010 (6th event)<br>111: Generate strobe on GLDCNT = 011 (7th event)<br>Reset type: SYSRSn |
| 6 | RESERVED | R-0 | 0h | Reserved |
| 5 | OSHTMODE | R/W | 0h | One Shot Load Mode Control Bit<br>0: One shot load mode is disabled and shadow to active loading happens continuously on all the chosen load strobes.<br>1: One shot mode is active. All load strobes are blocked until GLDCTL2[OSHTLD] is written with 1.<br>Note: One Shot mode can only be used with global shadow to active load mode enabled (GLDCTL[GLD]=1)<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-41. GLDCTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 4-1 | GLDMODE | R/W | 0h | Global Load Pulse selection for Shadow to Active Mode Reloads<br>0000: Load on Counter = 0 (CNT_ZRO)<br>0001: Load on Counter = Period (PRD_EQ)<br>0010: Load on either Counter = 0, or Counter = Period<br>0011: Load on SYNCEVT - this is logical OR of DCAEVT1.sync, DCBEVT1.sync, EPWMxSYNCI and TBCTL[SWFSYNC]<br>0100: Load on SYNCEVT or CNT_ZRO<br>0101: Load on SYNCEVT or PRD_EQ<br>0110: Load on SYNCEVT or CNT_ZRO or PRD_EQ<br>1000: Reserved<br>...<br>1110: Reserved<br>1111: Load on GLDCTL2[GFRCLD] write<br>Reset type: SYSRSn |
| 0 | GLD | R/W | 0h | Global Shadow to Active Load Event Control<br>0: Shadow to active reload for all shadowed registers happens as per the individual reload control bits specified (Compatible with previous EPWM versions).<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.21 GLDCFG Register (Offset = 35h) [Reset = 0000h]

GLDCFG is shown in Figure 15-114 and described in Table 15-42.

Return to the Summary Table.

Global PWM Load Config Register

**Figure 15-114. GLDCFG Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | AQCSFRC | AQCTLB_AQCTLB2 | AQCTLA_AQCTLA2 |
| R-0-0h | | | | | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DBCTL | DBFED_DBFEDHR | DBRED_DBREDHR | CMPD | CMPC | CMPB_CMPBHR | CMPA_CMPAHR | TBPRD_TBPRDHR |
| 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 15-42. 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<br>0: Registers use local reload configuration even if GLDCTL(GLD)=1 (reload is compatible with previous EPWMs)<br>1: Registers use global load configuration if this bit is set and GLDCTL(GLD)=1<br>Reset type: SYSRSn |
| 9 | AQCTLB_AQCTLB2 | R/W | 0h | Global load event configuration for AQCTLB_AQCTLB2<br>0: Registers use local reload configuration even if GLDCTL(GLD)=1 (reload is compatible with previous EPWMs)<br>1: Registers use global load configuration if this bit is set and GLDCTL(GLD)=1<br>Reset type: SYSRSn |
| 8 | AQCTLA_AQCTLA2 | R/W | 0h | Global load event configuration for AQCTLA_AQCTLA2<br>0: Registers use local reload configuration even if GLDCTL(GLD)=1 (reload is compatible with previous EPWMs)<br>1: Registers use global load configuration if this bit is set and GLDCTL(GLD)=1<br>Reset type: SYSRSn |
| 7 | DBCTL | R/W | 0h | Global load event configuration for DBCTL<br>0: Registers use local reload configuration even if GLDCTL(GLD)=1 (reload is compatible with previous EPWMs)<br>1: Registers use global load configuration if this bit is set and GLDCTL(GLD)=1<br>Reset type: SYSRSn |
| 6 | DBFED_DBFEDHR | R/W | 0h | Global load event configuration for DBFED_DBFEDHR<br>0: Registers use local reload configuration even if GLDCTL(GLD)=1 (reload is compatible with previous EPWMs)<br>1: Registers use global load configuration if this bit is set and GLDCTL(GLD)=1<br>Reset type: SYSRSn |
| 5 | DBRED_DBREDHR | R/W | 0h | Global load event configuration for DBRED_DBREDHR<br>0: Registers use local reload configuration even if GLDCTL(GLD)=1 (reload is compatible with previous EPWMs)<br>1: Registers use global load configuration if this bit is set and GLDCTL(GLD)=1<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-42. GLDCFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 4 | CMPD | R/W | 0h | Global load event configuration for CMPD<br>0: Registers use local reload configuration even if GLDCTL(GLD)=1 (reload is compatible with previous EPWMs)<br>1: Registers use global load configuration if this bit is set and GLDCTL(GLD)=1<br>Reset type: SYSRSn |
| 3 | CMPC | R/W | 0h | Global load event configuration for CMPC<br>0: Registers use local reload configuration even if GLDCTL(GLD)=1 (reload is compatible with previous EPWMs)<br>1: Registers use global load configuration if this bit is set and GLDCTL(GLD)=1<br>Reset type: SYSRSn |
| 2 | CMPB_CMPBHR | R/W | 0h | Global load event configuration for CMPB_CMPBHR<br>0: Registers use local reload configuration even if GLDCTL(GLD)=1 (reload is compatible with previous EPWMs)<br>1: Registers use global load configuration if this bit is set and GLDCTL(GLD)=1<br>Reset type: SYSRSn |
| 1 | CMPA_CMPAHR | R/W | 0h | Global load event configuration for CMPA_CMPAHR<br>0: Registers use local reload configuration even if GLDCTL(GLD)=1 (reload is compatible with previous EPWMs)<br>1: Registers use global load configuration if this bit is set and GLDCTL(GLD)=1<br>Reset type: SYSRSn |
| 0 | TBPRD_TBPRDHR | R/W | 0h | Global load event configuration for TBPRD_TBPRDHR<br>0: Registers use local reload configuration even if GLDCTL(GLD)=1 (reload is compatible with previous EPWMs)<br>1: Registers use global load configuration if this bit is set and GLDCTL(GLD)=1<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.22 EPWMXLINK Register (Offset = 38h) [Reset = 000XXXXXh]

EPWMXLINK is shown in Figure 15-115 and described in Table 15-43.

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 15-115. 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 15-43. EPWMXLINK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-28 | GLDCTL2LINK | R/W | 0h | GLDCTL2 Link Bits<br>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.<br>0000: ePWM1<br>0001: ePWM2<br>...<br>Up to the last instance of ePWM. All others are reserved.<br>Reset type: SYSRSn |
| 27-20 | RESERVED | R-0 | 0h | Reserved |
| 19-16 | CMPDLINK | R/W | X | CMPD Link Bits<br>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.<br>0000: ePWM1<br>0001: ePWM2<br>...<br>Up to the last instance of ePWM. All others are reserved.<br>Reset type: SYSRSn |
| 15-12 | CMPCLINK | R/W | X | CMPC Link Bits<br>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.<br>0000: ePWM1<br>0001: ePWM2<br>...<br>Up to the last instance of ePWM. All others are reserved.<br>Reset type: SYSRSn |
| 11-8 | CMPBLINK | R/W | X | CMPB_CMPBHR Link Bits<br>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.<br>0000: ePWM1<br>0001: ePWM2<br>...<br>Up to the last instance of ePWM. All others are reserved.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-43. EPWMXLINK Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 7-4 | CMPALINK | R/W | X | CMPA_CMPAHR Link Bits<br>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.<br>0000: ePWM1<br>0001: ePWM2<br>...<br>Up to the last instance of ePWM. All others are reserved.<br>Reset type: SYSRSn |
| 3-0 | TBPRDLINK | R/W | X | TBPRD_TBPRDHR Link Bits<br>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.<br>0000: ePWM1<br>0001: ePWM2<br>...<br>Up to the last instance of ePWM. All others are reserved.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.23 AQCTLA Register (Offset = 40h) [Reset = 0000h]

AQCTLA is shown in Figure 15-116 and described in Table 15-44.

Return to the Summary Table.

Action Qualifier Control Register For Output A

**Figure 15-116. AQCTLA Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | CBD | | CBU | |
| R-0-0h | | | | R/W-0h | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CAD | | CAU | | PRD | | ZRO | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 15-44. 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<br>Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.<br>00: Do nothing (action disabled)<br>01: Clear: force EPWMxA output low.<br>10: Set: force EPWMxA output high.<br>11: Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.<br>Reset type: SYSRSn |
| 9-8 | CBU | R/W | 0h | Action When TBCTR = CMPB on Up Count<br>Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.<br>00: Do nothing (action disabled)<br>01: Clear: force EPWMxA output low.<br>10: Set: force EPWMxA output high.<br>11: Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.<br>Reset type: SYSRSn |
| 7-6 | CAD | R/W | 0h | Action When TBCTR = CMPA on Down Count<br>Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.<br>00: Do nothing (action disabled)<br>01: Clear: force EPWMxA output low.<br>10: Set: force EPWMxA output high.<br>11: Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.<br>Reset type: SYSRSn |
| 5-4 | CAU | R/W | 0h | Action When TBCTR = CMPA on Up Count<br>Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.<br>00: Do nothing (action disabled)<br>01: Clear: force EPWMxA output low.<br>10: Set: force EPWMxA output high.<br>11: Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-44. AQCTLA Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-2 | PRD | R/W | 0h | Action When TBCTR = TBPRD<br>Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.<br>00: Do nothing (action disabled)<br>01: Clear: force EPWMxA output low.<br>10: Set: force EPWMxA output high.<br>11: Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.<br>Reset type: SYSRSn |
| 1-0 | ZRO | R/W | 0h | Action When TBCTR = 0<br>Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.<br>00: Do nothing (action disabled)<br>01: Clear: force EPWMxA output low.<br>10: Set: force EPWMxA output high.<br>11: Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.24 AQCTLA2 Register (Offset = 41h) [Reset = 0000h]

AQCTLA2 is shown in Figure 15-117 and described in Table 15-45.

Return to the Summary Table.

Additional Action Qualifier Control Register For Output A

**Figure 15-117. AQCTLA2 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| T2D | | T2U | | T1D | | T1U | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 15-45. 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<br>Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.<br>00: Do nothing (action disabled)<br>01: Clear: force EPWMxA output low.<br>10: Set: force EPWMxA output high.<br>11: Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.<br>Reset type: SYSRSn |
| 5-4 | T2U | R/W | 0h | Action when event occurs on T2 in UP-Count<br>Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.<br>00: Do nothing (action disabled)<br>01: Clear: force EPWMxA output low.<br>10: Set: force EPWMxA output high.<br>11: Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.<br>Reset type: SYSRSn |
| 3-2 | T1D | R/W | 0h | Action when event occurs on T1 in DOWN-Count<br>Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.<br>00: Do nothing (action disabled)<br>01: Clear: force EPWMxA output low.<br>10: Set: force EPWMxA output high.<br>11: Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.<br>Reset type: SYSRSn |
| 1-0 | T1U | R/W | 0h | Action when event occurs on T1 in UP-Count<br>Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.<br>00: Do nothing (action disabled)<br>01: Clear: force EPWMxA output low.<br>10: Set: force EPWMxA output high.<br>11: Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.25 AQCTLB Register (Offset = 42h) [Reset = 0000h]

AQCTLB is shown in Figure 15-118 and described in Table 15-46.

Return to the Summary Table.

Action Qualifier Control Register For Output B

**Figure 15-118. AQCTLB Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | CBD | | CBU | |
| R-0-0h | | | | R/W-0h | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CAD | | CAU | | PRD | | ZRO | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 15-46. 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<br>Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.<br>00: Do nothing (action disabled)<br>01: Clear: force EPWMxB output low.<br>10: Set: force EPWMxB output high.<br>11: Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low.<br>Reset type: SYSRSn |
| 9-8 | CBU | R/W | 0h | Action When TBCTR = CMPB on Up Count<br>Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.<br>00: Do nothing (action disabled)<br>01: Clear: force EPWMxB output low.<br>10: Set: force EPWMxB output high.<br>11: Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low.<br>Reset type: SYSRSn |
| 7-6 | CAD | R/W | 0h | Action When TBCTR = CMPA on Down Count<br>Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.<br>00: Do nothing (action disabled)<br>01: Clear: force EPWMxB output low.<br>10: Set: force EPWMxB output high.<br>11: Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low.<br>Reset type: SYSRSn |
| 5-4 | CAU | R/W | 0h | Action When TBCTR = CMPA on Up Count<br>Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.<br>00: Do nothing (action disabled)<br>01: Clear: force EPWMxB output low.<br>10: Set: force EPWMxB output high.<br>11: Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-46. AQCTLB Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-2 | PRD | R/W | 0h | Action When TBCTR = TBPRD<br>Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.<br>00: Do nothing (action disabled)<br>01: Clear: force EPWMxB output low.<br>10: Set: force EPWMxB output high.<br>11: Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low.<br>Reset type: SYSRSn |
| 1-0 | ZRO | R/W | 0h | Action When TBCTR = 0<br>Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.<br>00: Do nothing (action disabled)<br>01: Clear: force EPWMxB output low.<br>10: Set: force EPWMxB output high.<br>11: Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.26 AQCTLB2 Register (Offset = 43h) [Reset = 0000h]

AQCTLB2 is shown in Figure 15-119 and described in Table 15-47.

Return to the Summary Table.

Additional Action Qualifier Control Register For Output B

**Figure 15-119. AQCTLB2 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| T2D | | T2U | | T1D | | T1U | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 15-47. 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<br>Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.<br>00: Do nothing (action disabled)<br>01: Clear: force EPWMxB output low.<br>10: Set: force EPWMxB output high.<br>11: Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low.<br>Reset type: SYSRSn |
| 5-4 | T2U | R/W | 0h | Action when event occurs on T2 in UP-Count<br>Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.<br>00: Do nothing (action disabled)<br>01: Clear: force EPWMxB output low.<br>10: Set: force EPWMxB output high.<br>11: Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low.<br>Reset type: SYSRSn |
| 3-2 | T1D | R/W | 0h | Action when event occurs on T1 in DOWN-Count<br>Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.<br>00: Do nothing (action disabled)<br>01: Clear: force EPWMxB output low.<br>10: Set: force EPWMxB output high.<br>11: Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low.<br>Reset type: SYSRSn |
| 1-0 | T1U | R/W | 0h | Action when event occurs on T1 in UP-Count<br>Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.<br>00: Do nothing (action disabled)<br>01: Clear: force EPWMxB output low.<br>10: Set: force EPWMxB output high.<br>11: Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.27 AQSFRC Register (Offset = 47h) [Reset = 0000h]

AQSFRC is shown in Figure 15-120 and described in Table 15-48.

Return to the Summary Table.

Action Qualifier Software Force Register

**Figure 15-120. AQSFRC Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RLDCSF | | OTSFB | ACTSFB | | OTSFA | ACTSFA | |
| R/W-0h | | R-0/W1S-0h | R/W-0h | | R-0/W1S-0h | R/W-0h | |

**Table 15-48. AQSFRC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R-0 | 0h | Reserved |
| 7-6 | RLDCSF | R/W | 0h | AQCSFRC Active Register Reload From Shadow Options<br>00: Load on time-base counter equals zero<br>01: Load on time-base counter equals period<br>10: Load on time-base counter equals zero or counter equals period<br>11: Load immediately (the active register is directly accessed by the CPU and is not loaded from the shadow register).<br>Reset type: SYSRSn |
| 5 | OTSFB | R-0/W1S | 0h | One-Time Software Forced Event on Output B<br>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.<br>1: Initiates a single software forced event<br>Reset type: SYSRSn |
| 4-3 | ACTSFB | R/W | 0h | Action When One-Time Software Force B is Invoked<br>00: Does nothing (action disabled)<br>01: Clear (low)<br>10: Set (high)<br>11: Toggle (Low -> High, High -> Low)<br>Note: This action is not qualified by counter direction (CNT_dir)<br>Reset type: SYSRSn |
| 2 | OTSFA | R-0/W1S | 0h | One-Time Software Forced Event on Output A<br>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.<br>1: Initiates a single software forced event<br>Reset type: SYSRSn |
| 1-0 | ACTSFA | R/W | 0h | Action When One-Time Software Force A Is Invoked<br>00: Does nothing (action disabled)<br>01: Clear (low)<br>10: Set (high)<br>11: Toggle (Low -> High, High -> Low)<br>Note: This action is not qualified by counter direction (CNT_dir)<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.28 AQCSFRC Register (Offset = 49h) [Reset = 0000h]

AQCSFRC is shown in Figure 15-121 and described in Table 15-49.

Return to the Summary Table.

Action Qualifier Continuous S/W Force Register

**Figure 15-121. AQCSFRC Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | CSFB | | CSFA | |
| R-0-0h | | | | R/W-0h | | R/W-0h | |

**Table 15-49. 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<br>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].<br>00: Software forcing is disabled and has no effect<br>01: Forces a continuous low on output B<br>10: Forces a continuous high on output B<br>11: Software forcing is disabled and has no effect<br>Reset type: SYSRSn |
| 1-0 | CSFA | R/W | 0h | Continuous Software Force on Output A<br>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.<br>00: Software forcing is disabled and has no effect<br>01: Forces a continuous low on output A<br>10: Forces a continuous high on output A<br>11: Software forcing is disabled and has no effect<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.29 DBREDHR Register (Offset = 50h) [Reset = 0000h]

DBREDHR is shown in Figure 15-122 and described in Table 15-50.

Return to the Summary Table.

Dead-Band Generator Rising Edge Delay High Resolution Register

**Figure 15-122. DBREDHR Register**

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

**Table 15-50. DBREDHR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-9 | DBREDHR | R/W | 0h | Dead Band Rising Edge Delay High Resolution Bits<br>Reset type: SYSRSn |
| 8 | RESERVED | R | 0h | Reserved |
| 7-1 | RESERVED | R | 0h | Reserved |
| 0 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.30 DBRED Register (Offset = 51h) [Reset = 0000h]

DBRED is shown in Figure 15-123 and described in Table 15-51.

Return to the Summary Table.

Dead-Band Generator Rising Edge Delay Count Register

**Figure 15-123. DBRED Register**

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

**Table 15-51. 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<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.31 DBFEDHR Register (Offset = 52h) [Reset = 0000h]

DBFEDHR is shown in Figure 15-124 and described in Table 15-52.

Return to the Summary Table.

Dead-Band Generator Falling Edge Delay High Resolution Register

**Figure 15-124. DBFEDHR Register**

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

**Table 15-52. DBFEDHR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-9 | DBFEDHR | R/W | 0h | Dead Band Falling Edge Delay High Resolution Bits<br>Reset type: SYSRSn |
| 8 | RESERVED | R | 0h | Reserved |
| 7-1 | RESERVED | R | 0h | Reserved |
| 0 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.32 DBFED Register (Offset = 53h) [Reset = 0000h]

DBFED is shown in Figure 15-125 and described in Table 15-53.

Return to the Summary Table.

Dead-Band Generator Falling Edge Delay Count Register

**Figure 15-125. DBFED Register**

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

**Table 15-53. 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<br>14-bit counter<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.33 TBPHS Register (Offset = 60h) [Reset = 00000000h]

TBPHS is shown in Figure 15-126 and described in Table 15-54.

Return to the Summary Table.

Time Base Phase Register

**Figure 15-126. TBPHS Register**

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

**Table 15-54. TBPHS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | TBPHS | R/W | 0h | Phase Offset Register<br>These bits set time-base counter phase of the selected ePWM relative to the time-base that is supplying the synchronization input signal.<br>- If TBCTL[PHSEN] = 0, then the synchronization event is ignored and the time-base counter is not loaded with the phase.<br>- 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.<br>Reset type: SYSRSn |
| 15-0 | TBPHSHR | R/W | 0h | Phase Offset (High Resolution) Register.<br>TBPHSHR must not be used. Instead TRREM (HRPWM remainder register) must be used to mimic the functionality of TBPHSHR.<br>The lower 8 bits in this register are ignored - writes are ignored and reads return zero<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.34 TBPRDHR Register (Offset = 62h) [Reset = 0000h]

TBPRDHR is shown in Figure 15-127 and described in Table 15-55.

Return to the Summary Table.

Time Base Period High Resolution Register

**Figure 15-127. TBPRDHR Register**

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

**Table 15-55. TBPRDHR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | TBPRDHR | R/W | 0h | Period High Resolution Bits<br>The upper 8-bits contain the high-resolution portion of the period value. The TBPRDHR register is not affected by the TBCTL[PRDLD] bit. Reads from this register always reflect the shadow register. Likewise writes are also to the shadow register. The TBPRDHR register is only used when the high resolution period feature is enabled. This register is only available with ePWM modules which support high-resolution period control.<br>The lower 8 bits in this register are ignored - writes are ignored and reads return zero<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.35 TBPRD Register (Offset = 63h) [Reset = 0000h]

TBPRD is shown in Figure 15-128 and described in Table 15-56.

Return to the Summary Table.

Time Base Period Register

**Figure 15-128. TBPRD Register**

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

**Table 15-56. TBPRD Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | TBPRD | R/W | 0h | Time Base Period Register<br>These bits determine the period of the time-base counter. This sets the PWM frequency. Shadowing of this register is enabled and disabled by the TBCTL[PRDLD] bit. By default this register is shadowed.<br>- If TBCTL[PRDLD] = 0, then the shadow is enabled and any write or read will automatically go to the shadow register. In this case, the active register will be loaded from the shadow register when the time-base counter equals zero.<br>- If TBCTL[PRDLD] = 1, then the shadow is disabled and any write or read will go directly to the active register, that is the register actively controlling the hardware.<br>- The active and shadow registers share the same memory map address.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.36 CMPA Register (Offset = 6Ah) [Reset = 00000000h]

CMPA is shown in Figure 15-129 and described in Table 15-57.

Return to the Summary Table.

Counter Compare A Register

**Figure 15-129. CMPA Register**

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

**Table 15-57. CMPA Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | CMPA | R/W | 0h | Compare A Register<br>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:<br>- Do nothing<br>the event is ignored.<br>- Clear: Pull the EPWMxA and/or EPWMxB signal low<br>- Set: Pull the EPWMxA and/or EPWMxB signal high<br>- Toggle the EPWMxA and/or EPWMxB signal<br>Shadowing of this register is enabled and disabled by the CMPCTL[SHDWAMODE] bit. By default this register is shadowed.<br>- 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.<br>- Before a write, the CMPCTL[SHDWAFULL] bit can be read to determine if the shadow register is currently full.<br>- 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.<br>- In either mode, the active and shadow registers share the same memory map address.<br>Reset type: SYSRSn |
| 15-0 | CMPAHR | R/W | 0h | Compare A HRPWM Extension Register<br>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.<br>The lower 8 bits in this register are ignored<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.37 CMPB Register (Offset = 6Ch) [Reset = 00000000h]

CMPB is shown in Figure 15-130 and described in Table 15-58.

Return to the Summary Table.

Compare B Register

**Figure 15-130. CMPB Register**

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

**Table 15-58. CMPB Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | CMPB | R/W | 0h | Compare B Register<br>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:<br>- Do nothing<br>the event is ignored.<br>- Clear: Pull the EPWMxA and/or EPWMxB signal low<br>- Set: Pull the EPWMxA and/or EPWMxB signal high<br>- Toggle the EPWMxA and/or EPWMxB signal<br>Shadowing of this register is enabled and disabled by the CMPCTL[SHDWBMODE] bit. By default this register is shadowed.<br>- 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.<br>- Before a write, the CMPCTL[SHDWBFULL] bit can be read to determine if the shadow register is currently full.<br>- 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.<br>- In either mode, the active and shadow registers share the same memory map address.<br>Reset type: SYSRSn |
| 15-0 | CMPBHR | R/W | 0h | Compare B High Resolution Bits<br>The lower 8 bits in this register are ignored<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.38 CMPC Register (Offset = 6Fh) [Reset = 0000h]

CMPC is shown in Figure 15-131 and described in Table 15-59.

Return to the Summary Table.

Counter Compare C Register
LINK feature access should always be 16-bit

**Figure 15-131. CMPC Register**

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

**Table 15-59. CMPC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | CMPC | R/W | 0h | Compare C Register<br>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.<br>Shadowing of this register is enabled and disabled by the CMPCTL2[SHDWCMODE] bit. By default this register is shadowed.<br>- 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:<br>- 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.<br>- In either mode, the active and shadow registers share the same memory map address.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.39 CMPD Register (Offset = 71h) [Reset = 0000h]

CMPD is shown in Figure 15-132 and described in Table 15-60.

Return to the Summary Table.

Counter Compare D Register
LINK feature access should always be 16-bit

**Figure 15-132. CMPD Register**

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

**Table 15-60. CMPD Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | CMPD | R/W | 0h | Compare D Register<br>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.<br>Shadowing of this register is enabled and disabled by the CMPCTL2[SHDWDMODE] bit. By default this register is shadowed.<br>- 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:<br>- 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.<br>- In either mode, the active and shadow registers share the same memory map address.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.40 GLDCTL2 Register (Offset = 74h) [Reset = 0000h]

GLDCTL2 is shown in Figure 15-133 and described in Table 15-61.

Return to the Summary Table.

Global PWM Load Control Register 2

**Figure 15-133. GLDCTL2 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | GFRCLD | OSHTLD |
| R-0-0h | | | | | | R-0/W1S-0h | R-0/W1S-0h |

**Table 15-61. 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<br>0: Writing of 0 will be ignored. Always reads back a 0.<br>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.<br>Reset type: SYSRSn |
| 0 | OSHTLD | R-0/W1S | 0h | Enable Reload Event in One Shot Mode<br>0: Writing of 0 will be ignored. Always reads back a 0.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.41 SWVDELVAL Register (Offset = 77h) [Reset = 0000h]

SWVDELVAL is shown in Figure 15-134 and described in Table 15-62.

Return to the Summary Table.

Software Valley Mode Delay Register

**Figure 15-134. 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 15-62. SWVDELVAL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | SWVDELVAL | R/W | 0h | Software Valley Delay Value Register<br>This register can be optionally used define offset value for the hardware calculated delay HWDELAYVAL as defined in VCAPCTL[VDELAYDIV] bits.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.42 TZSEL Register (Offset = 80h) [Reset = 0000h]

TZSEL is shown in Figure 15-135 and described in Table 15-63.

Return to the Summary Table.

Trip Zone Select Register

**Figure 15-135. TZSEL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| DCBEVT1 | DCAEVT1 | OSHT6 | OSHT5 | OSHT4 | OSHT3 | OSHT2 | OSHT1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DCBEVT2 | DCAEVT2 | CBC6 | CBC5 | CBC4 | CBC3 | CBC2 | CBC1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 15-63. TZSEL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | DCBEVT1 | R/W | 0h | Digital Compare Output B Event 1 Select<br>0: Disable DCBEVT1 as one-shot-trip source for this ePWM module.<br>1: Enable DCBEVT1 as one-shot-trip source for this ePWM module.<br>Reset type: SYSRSn |
| 14 | DCAEVT1 | R/W | 0h | Digital Compare Output A Event 1 Select<br>0: Disable DCAEVT1 as one-shot-trip source for this ePWM module.<br>1: Enable DCAEVT1 as one-shot-trip source for this ePWM module.<br>Reset type: SYSRSn |
| 13 | OSHT6 | R/W | 0h | Trip-zone 6 (TZ6) Select<br>0: Disable TZ6 as a one-shot trip source for this ePWM module<br>1: Enable TZ6 as a one-shot trip source for this ePWM module<br>Reset type: SYSRSn |
| 12 | OSHT5 | R/W | 0h | Trip-zone 5 (TZ5) Select<br>0: Disable TZ5 as a one-shot trip source for this ePWM module<br>1: Enable TZ5 as a one-shot trip source for this ePWM module<br>Reset type: SYSRSn |
| 11 | OSHT4 | R/W | 0h | Trip-zone 4 (TZ4) Select<br>0: Disable TZ4 as a one-shot trip source for this ePWM module<br>1: Enable TZ4 as a one-shot trip source for this ePWM module<br>Reset type: SYSRSn |
| 10 | OSHT3 | R/W | 0h | Trip-zone 3 (TZ3) Select<br>0: Disable TZ3 as a one-shot trip source for this ePWM module<br>1: Enable TZ3 as a one-shot trip source for this ePWM module<br>Reset type: SYSRSn |
| 9 | OSHT2 | R/W | 0h | Trip-zone 2 (TZ2) Select<br>0: Disable TZ2 as a one-shot trip source for this ePWM module<br>1: Enable TZ2 as a one-shot trip source for this ePWM module<br>Reset type: SYSRSn |
| 8 | OSHT1 | R/W | 0h | Trip-zone 1 (TZ1) Select<br>0: Disable TZ1 as a one-shot trip source for this ePWM module<br>1: Enable TZ1 as a one-shot trip source for this ePWM module<br>Reset type: SYSRSn |
| 7 | DCBEVT2 | R/W | 0h | Digital Compare Output B Event 2 Select<br>0: Disable DCBEVT2 as a CBC trip source for this ePWM module<br>1: Enable DCBEVT2 as a CBC trip source for this ePWM module<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-63. TZSEL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 6 | DCAEVT2 | R/W | 0h | Digital Compare Output A Event 2 Select<br>0: Disable DCAEVT2 as a CBC trip source for this ePWM module<br>1: Enable DCAEVT2 as a CBC trip source for this ePWM module<br>Reset type: SYSRSn |
| 5 | CBC6 | R/W | 0h | Trip-zone 6 (TZ6) Select<br>0: Disable TZ6 as a CBC trip source for this ePWM module<br>1: Enable TZ6 as a CBC trip source for this ePWM module<br>Reset type: SYSRSn |
| 4 | CBC5 | R/W | 0h | Trip-zone 5 (TZ5) Select<br>0: Disable TZ5 as a CBC trip source for this ePWM module<br>1: Enable TZ5 as a CBC trip source for this ePWM module<br>Reset type: SYSRSn |
| 3 | CBC4 | R/W | 0h | Trip-zone 4 (TZ4) Select<br>0: Disable TZ4 as a CBC trip source for this ePWM module<br>1: Enable TZ4 as a CBC trip source for this ePWM module<br>Reset type: SYSRSn |
| 2 | CBC3 | R/W | 0h | Trip-zone 3 (TZ3) Select<br>0: Disable TZ3 as a CBC trip source for this ePWM module<br>1: Enable TZ3 as a CBC trip source for this ePWM module<br>Reset type: SYSRSn |
| 1 | CBC2 | R/W | 0h | Trip-zone 2 (TZ2) Select<br>0: Disable TZ2 as a CBC trip source for this ePWM module<br>1: Enable TZ2 as a CBC trip source for this ePWM module<br>Reset type: SYSRSn |
| 0 | CBC1 | R/W | 0h | Trip-zone 1 (TZ1) Select<br>0: Disable TZ1 as a CBC trip source for this ePWM module<br>1: Enable TZ1 as a CBC trip source for this ePWM module<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.43 TZDCSEL Register (Offset = 82h) [Reset = 0000h]

TZDCSEL is shown in Figure 15-136 and described in Table 15-64.

Return to the Summary Table.

Trip Zone Digital Comparator Select Register

**Figure 15-136. TZDCSEL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | DCBEVT2 | | | DCBEVT1 |
| R-0-0h | | | | R/W-0h | | | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DCBEVT1 | | DCAEVT2 | | | DCAEVT1 | | |
| R/W-0h | | R/W-0h | | | R/W-0h | | |

**Table 15-64. 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<br>000: Event disabled<br>001: DCBH = low, DCBL = don't care<br>010: DCBH = high, DCBL = don't care<br>011: DCBL = low, DCBH = don't care<br>100: DCBL = high, DCBH = don't care<br>101: DCBL = high, DCBH = low<br>110: Reserved<br>111: Reserved<br>Reset type: SYSRSn |
| 8-6 | DCBEVT1 | R/W | 0h | Digital Compare Output B Event 1 Selection<br>000: Event disabled<br>001: DCBH = low, DCBL = don't care<br>010: DCBH = high, DCBL = don't care<br>011: DCBL = low, DCBH = don't care<br>100: DCBL = high, DCBH = don't care<br>101: DCBL = high, DCBH = low<br>110: Reserved<br>111: Reserved<br>Reset type: SYSRSn |
| 5-3 | DCAEVT2 | R/W | 0h | Digital Compare Output A Event 2 Selection<br>000: Event disabled<br>001: DCAH = low, DCAL = don't care<br>010: DCAH = high, DCAL = don't care<br>011: DCAL = low, DCAH = don't care<br>100: DCAL = high, DCAH = don't care<br>101: DCAL = high, DCAH = low<br>110: Reserved<br>111: Reserved<br>Reset type: SYSRSn |
| 2-0 | DCAEVT1 | R/W | 0h | Digital Compare Output A Event 1 Selection<br>000: Event disabled<br>001: DCAH = low, DCAL = don't care<br>010: DCAH = high, DCAL = don't care<br>011: DCAL = low, DCAH = don't care<br>100: DCAL = high, DCAH = don't care<br>101: DCAL = high, DCAH = low<br>110: Reserved<br>111: Reserved<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.44 TZCTL Register (Offset = 84h) [Reset = 0000h]

TZCTL is shown in Figure 15-137 and described in Table 15-65.

Return to the Summary Table.

Trip Zone Control Register

**Figure 15-137. TZCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | DCBEVT2 | | DCBEVT1 | |
| R-0-0h | | | | R/W-0h | | R/W-0h | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| DCAEVT2 | | DCAEVT1 | | TZB | | TZA | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 15-65. 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<br>00: High-impedance (EPWMxB = High-impedance state)<br>01: Force EPWMxB to a high state.<br>10: Force EPWMxB to a low state.<br>11: Do Nothing, trip action is disabled<br>Reset type: SYSRSn |
| 9-8 | DCBEVT1 | R/W | 0h | Digital Compare Output B Event 1 Action On EPWMxB<br>00: High-impedance (EPWMxB = High-impedance state)<br>01: Force EPWMxB to a high state.<br>10: Force EPWMxB to a low state.<br>11: Do Nothing, trip action is disabled<br>Reset type: SYSRSn |
| 7-6 | DCAEVT2 | R/W | 0h | Digital Compare Output A Event 2 Action On EPWMxA<br>00: High-impedance (EPWMxA = High-impedance state)<br>01: Force EPWMxA to a high state.<br>10: Force EPWMxA to a low state.<br>11: Do Nothing, trip action is disabled<br>Reset type: SYSRSn |
| 5-4 | DCAEVT1 | R/W | 0h | Digital Compare Output A Event 1 Action On EPWMxA<br>00: High-impedance (EPWMxA = High-impedance state)<br>01: Force EPWMxA to a high state.<br>10: Force EPWMxA to a low state.<br>11: Do Nothing, trip action is disabled<br>Reset type: SYSRSn |
| 3-2 | TZB | R/W | 0h | TZ1 to TZ6, DCAEVT1/2, DCBEVT1/2Trip Action On EPWMxB<br>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.<br>00: High-impedance (EPWMxB = High-impedance state)<br>01: Force EPWMxB to a high state<br>10: Force EPWMxB to a low state<br>11: Do nothing, no action is taken on EPWMxB.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-65. 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<br>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.<br>00: High-impedance (EPWMxA = High-impedance state)<br>01: Force EPWMxA to a high state<br>10: Force EPWMxA to a low state<br>11: Do nothing, no action is taken on EPWMxA.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.45 TZCTL2 Register (Offset = 85h) [Reset = 0000h]

TZCTL2 is shown in Figure 15-138 and described in Table 15-66.

Return to the Summary Table.

Additional Trip Zone Control Register

**Figure 15-138. TZCTL2 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| ETZE | RESERVED | | | TZBD | | | TZBU |
| R/W-0h | R-0-0h | | | R/W-0h | | | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TZBU | | TZAD | | | TZAU | | |
| R/W-0h | | R/W-0h | | | R/W-0h | | |

**Table 15-66. TZCTL2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | ETZE | R/W | 0h | TZCTL2 Enable<br>0: Use trip action from TZCTL (legacy EPWM compatibility)<br>1: Use trip action defined in TZCTL2, TZCTLDCA and TZCTLDCB. Settings in TZCTL are ignored<br>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<br>000: HiZ (EPWMxB = HiZ state)<br>001: Forced Hi (EPWMxB = High state)<br>010: Forced Lo (EPWMxB = Lo state)<br>011: Toggle (Low -> High, High -> Low)<br>100: Reserved<br>101: Reserved<br>110: Reserved<br>111: Do Nothing, trip action is disabled<br>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<br>000: HiZ (EPWMxB = HiZ state)<br>001: Forced Hi (EPWMxB = High state)<br>010: Forced Lo (EPWMxB = Lo state)<br>011: Toggle (Low -> High, High -> Low)<br>100: Reserved<br>101: Reserved<br>110: Reserved<br>111: Do Nothing, trip action is disabled<br>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<br>000: HiZ (EPWMxA = HiZ state)<br>001: Forced Hi (EPWMxA = High state)<br>010: Forced Lo (EPWMxA = Lo state)<br>011: Toggle (Low -> High, High -> Low)<br>100: Reserved<br>101: Reserved<br>110: Reserved<br>111: Do Nothing, trip action is disabled<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-66. 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<br>000: HiZ (EPWMxA = HiZ state)<br>001: Forced Hi (EPWMxA = High state)<br>010: Forced Lo (EPWMxA = Lo state)<br>011: Toggle (Low -> High, High -> Low)<br>100: Reserved<br>101: Reserved<br>110: Reserved<br>111: Do Nothing, trip action is disabled<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.46 TZCTLDCA Register (Offset = 86h) [Reset = 0000h]

TZCTLDCA is shown in Figure 15-139 and described in Table 15-67.

Return to the Summary Table.

Trip Zone Control Register Digital Compare A

**Figure 15-139. TZCTLDCA Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | DCAEVT2D | | | DCAEVT2U |
| R-0-0h | | | | R/W-0h | | | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DCAEVT2U | | DCAEVT1D | | | DCAEVT1U | | |
| R/W-0h | | R/W-0h | | | R/W-0h | | |

**Table 15-67. 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<br>000: HiZ (EPWMxA = HiZ state)<br>001: Forced Hi (EPWMxA = High state)<br>010: Forced Lo (EPWMxA = Lo state)<br>011: Toggle (Low -> High, High -> Low)<br>100: Reserved<br>101: Reserved<br>110: Reserved<br>111: Do Nothing, trip action is disabled<br>Reset type: SYSRSn |
| 8-6 | DCAEVT2U | R/W | 0h | Digital Compare Output A Event 2 Action On EPWMxA while Count direction is UP<br>000: HiZ (EPWMxA = HiZ state)<br>001: Forced Hi (EPWMxA = High state)<br>010: Forced Lo (EPWMxA = Lo state)<br>011: Toggle (Low -> High, High -> Low)<br>100: Reserved<br>101: Reserved<br>110: Reserved<br>111: Do Nothing, trip action is disabled<br>Reset type: SYSRSn |
| 5-3 | DCAEVT1D | R/W | 0h | Digital Compare Output A Event 1 Action On EPWMxA while Count direction is DOWN<br>000: HiZ (EPWMxA = HiZ state)<br>001: Forced Hi (EPWMxA = High state)<br>010: Forced Lo (EPWMxA = Lo state)<br>011: Toggle (Low -> High, High -> Low)<br>100: Reserved<br>101: Reserved<br>110: Reserved<br>111: Do Nothing, trip action is disabled<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-67. TZCTLDCA Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 2-0 | DCAEVT1U | R/W | 0h | Digital Compare Output A Event 1 Action On EPWMxA while Count direction is UP<br>000: HiZ (EPWMxA = HiZ state)<br>001: Forced Hi (EPWMxA = High state)<br>010: Forced Lo (EPWMxA = Lo state)<br>011: Toggle (Low -> High, High -> Low)<br>100: Reserved<br>101: Reserved<br>110: Reserved<br>111: Do Nothing, trip action is disabled<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.47 TZCTLDCB Register (Offset = 87h) [Reset = 0000h]

TZCTLDCB is shown in Figure 15-140 and described in Table 15-68.

Return to the Summary Table.

Trip Zone Control Register Digital Compare B

**Figure 15-140. TZCTLDCB Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | DCBEVT2D | | | DCBEVT2U |
| R-0-0h | | | | R/W-0h | | | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DCBEVT2U | | DCBEVT1D | | | DCBEVT1U | | |
| R/W-0h | | R/W-0h | | | R/W-0h | | |

**Table 15-68. 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<br>000: HiZ (EPWMxB = HiZ state)<br>001: Forced Hi (EPWMxB = High state)<br>010: Forced Lo (EPWMxB = Lo state)<br>011: Toggle (Low -> High, High -> Low)<br>100: Reserved<br>101: Reserved<br>110: Reserved<br>111: Do Nothing, trip action is disabled<br>Reset type: SYSRSn |
| 8-6 | DCBEVT2U | R/W | 0h | Digital Compare Output B Event 2 Action On EPWMxB while Count direction is UP<br>000: HiZ (EPWMxB = HiZ state)<br>001: Forced Hi (EPWMxB = High state)<br>010: Forced Lo (EPWMxB = Lo state)<br>011: Toggle (Low -> High, High -> Low)<br>100: Reserved<br>101: Reserved<br>110: Reserved<br>111: Do Nothing, trip action is disabled<br>Reset type: SYSRSn |
| 5-3 | DCBEVT1D | R/W | 0h | Digital Compare Output B Event 1 Action On EPWMxB while Count direction is DOWN<br>000: HiZ (EPWMxB = HiZ state)<br>001: Forced Hi (EPWMxB = High state)<br>010: Forced Lo (EPWMxB = Lo state)<br>011: Toggle (Low -> High, High -> Low)<br>100: Reserved<br>101: Reserved<br>110: Reserved<br>111: Do Nothing, trip action is disabled<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-68. 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<br>000: HiZ (EPWMxB = HiZ state)<br>001: Forced Hi (EPWMxB = High state)<br>010: Forced Lo (EPWMxB = Lo state)<br>011: Toggle (Low -> High, High -> Low)<br>100: Reserved<br>101: Reserved<br>110: Reserved<br>111: Do Nothing, trip action is disabled<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.48 TZEINT Register (Offset = 8Dh) [Reset = 0000h]

TZEINT is shown in Figure 15-141 and described in Table 15-69.

Return to the Summary Table.

Trip Zone Enable Interrupt Register

**Figure 15-141. TZEINT Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | DCBEVT2 | DCBEVT1 | DCAEVT2 | DCAEVT1 | OST | CBC | RESERVED |
| R-0-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R-0-0h |

**Table 15-69. 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<br>0: Disabled<br>1: Enabled<br>Reset type: SYSRSn |
| 5 | DCBEVT1 | R/W | 0h | Digital Compare Output B Event 1 Interrupt Enable<br>0: Disabled<br>1: Enabled<br>Reset type: SYSRSn |
| 4 | DCAEVT2 | R/W | 0h | Digital Compare Output A Event 2 Interrupt Enable<br>0: Disabled<br>1: Enabled<br>Reset type: SYSRSn |
| 3 | DCAEVT1 | R/W | 0h | Digital Compare Output A Event 1 Interrupt Enable<br>0: Disabled<br>1: Enabled<br>Reset type: SYSRSn |
| 2 | OST | R/W | 0h | Trip-zone One-Shot Interrupt Enable<br>0: Disable one-shot interrupt generation<br>1: Enable Interrupt generation<br>a one-shot trip event will cause a EPWMx_TZINT PIE interrupt.<br>Reset type: SYSRSn |
| 1 | CBC | R/W | 0h | Trip-zone Cycle-by-Cycle Interrupt Enable<br>0: Disable cycle-by-cycle interrupt generation.<br>1: Enable interrupt generation<br>a cycle-by-cycle trip event will cause an EPWMx_TZINT PIE interrupt.<br>Reset type: SYSRSn |
| 0 | RESERVED | R-0 | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.49 TZFLG Register (Offset = 93h) [Reset = 0000h]

TZFLG is shown in Figure 15-142 and described in Table 15-70.

Return to the Summary Table.

Trip Zone Flag Register

**Figure 15-142. TZFLG Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | DCBEVT2 | DCBEVT1 | DCAEVT2 | DCAEVT1 | OST | CBC | INT |
| R-0-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 15-70. 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<br>0: Indicates no trip event has occurred on DCBEVT2<br>1: Indicates a trip event has occurred for the event defined for DCBEVT2<br>Reset type: SYSRSn |
| 5 | DCBEVT1 | R | 0h | Latched Status Flag for Digital Compare Output B Event 1<br>0: Indicates no trip event has occurred on DCBEVT1<br>1: Indicates a trip event has occurred for the event defined for DCBEVT1<br>Reset type: SYSRSn |
| 4 | DCAEVT2 | R | 0h | Latched Status Flag for Digital Compare Output A Event 2<br>0: Indicates no trip event has occurred on DCAEVT2<br>1: Indicates a trip event has occurred for the event defined for DCAEVT2<br>Reset type: SYSRSn |
| 3 | DCAEVT1 | R | 0h | Latched Status Flag for Digital Compare Output A Event 1<br>0: Indicates no trip event has occurred on DCAEVT1<br>1: Indicates a trip event has occurred for the event defined for DCAEVT1<br>Reset type: SYSRSn |
| 2 | OST | R | 0h | Latched Status Flag for A One-Shot Trip Event<br>0: No one-shot trip event has occurred.<br>1: Indicates a trip event has occurred on a pin selected as a one-shot trip source.<br>This bit is cleared by writing the appropriate value to the TZCLR register.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-70. TZFLG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1 | CBC | R | 0h | Latched Status Flag for Cycle-By-Cycle Trip Event<br>0: No cycle-by-cycle trip event has occurred.<br>1: Indicates a trip event has occurred on a signal selected as a cycle-by-cycle trip source. The<br>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.<br>This bit is cleared by writing the appropriate value to the TZCLR register.<br>Reset type: SYSRSn |
| 0 | INT | R | 0h | Latched Trip Interrupt Status Flag<br>0: Indicates no interrupt has been generated.<br>1: Indicates an EPWMx_TZINT PIE interrupt was generated because of a trip condition.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.50 TZCBCFLG Register (Offset = 94h) [Reset = 0000h]

TZCBCFLG is shown in Figure 15-143 and described in Table 15-71.

Return to the Summary Table.

Trip Zone CBC Flag Register

**Figure 15-143. TZCBCFLG Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DCBEVT2 | DCAEVT2 | CBC6 | CBC5 | CBC4 | CBC3 | CBC2 | CBC1 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 15-71. 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<br>0: Reading a 0 indicates that no trip has occurred on DCBEVT2.<br>1: Reading a 1 indicates a trip has occured on the DCBEVT2 selected event.<br>Reset type: SYSRSn |
| 6 | DCAEVT2 | R | 0h | Latched Status Flag for Digital Compare A Output Event 2 Trip Latch<br>0: Reading a 0 indicates that no trip has occurred on DCAEVT2.<br>1: Reading a 1 indicates a trip has occured on the DCAEVT2 selected event.<br>Reset type: SYSRSn |
| 5 | CBC6 | R | 0h | Latched Status Flag for CBC6 Trip Latch<br>0: Reading a 0 indicates that no trip has occurred on CBC6.<br>1: Reading a 1 indicates a trip has occured on the CBC6 selected event.<br>Reset type: SYSRSn |
| 4 | CBC5 | R | 0h | Latched Status Flag for CBC5 Trip Latch<br>0: Reading a 0 indicates that no trip has occurred on CBC5.<br>1: Reading a 1 indicates a trip has occured on the CBC5 selected event.<br>Reset type: SYSRSn |
| 3 | CBC4 | R | 0h | Latched Status Flag for CBC4 Trip Latch<br>0: Reading a 0 indicates that no trip has occurred on CBC4.<br>1: Reading a 1 indicates a trip has occured on the CBC4 selected event.<br>Reset type: SYSRSn |
| 2 | CBC3 | R | 0h | Latched Status Flag for CBC3 Trip Latch<br>0: Reading a 0 indicates that no trip has occurred on CBC3.<br>1: Reading a 1 indicates a trip has occured on the CBC3 selected event.<br>Reset type: SYSRSn |
| 1 | CBC2 | R | 0h | Latched Status Flag for CBC2 Trip Latch<br>0: Reading a 0 indicates that no trip has occurred on CBC2.<br>1: Reading a 1 indicates a trip has occured on the CBC2 selected event.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-71. TZCBCFLG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | CBC1 | R | 0h | Latched Status Flag for CBC1 Trip Latch<br>0: Reading a 0 indicates that no trip has occurred on CBC1.<br>1: Reading a 1 indicates a trip has occured on the CBC1 selected event.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.51 TZOSTFLG Register (Offset = 95h) [Reset = 0000h]

TZOSTFLG is shown in Figure 15-144 and described in Table 15-72.

Return to the Summary Table.

Trip Zone OST Flag Register

**Figure 15-144. TZOSTFLG Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DCBEVT1 | DCAEVT1 | OST6 | OST5 | OST4 | OST3 | OST2 | OST1 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 15-72. 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<br>0: Reading a 0 indicates that no trip has occurred on DCBEVT1.<br>1: Reading a 1 indicates a trip has occured on the DCBEVT1 selected event.<br>Reset type: SYSRSn |
| 6 | DCAEVT1 | R | 0h | Latched Status Flag for Digital Compare A Output Event 1 Trip Latch<br>0: Reading a 0 indicates that no trip has occurred on DCAEVT1.<br>1: Reading a 1 indicates a trip has occured on the DCAEVT1 selected event.<br>Reset type: SYSRSn |
| 5 | OST6 | R | 0h | Latched Status Flag for OST6 Trip Latch<br>0: Reading a 0 indicates that no trip has occurred on OST6.<br>1: Reading a 1 indicates a trip has occured on the OST6 selected event.<br>Reset type: SYSRSn |
| 4 | OST5 | R | 0h | Latched Status Flag for OST5 Trip Latch<br>0: Reading a 0 indicates that no trip has occurred on OST5.<br>1: Reading a 1 indicates a trip has occured on the OST5 selected event.<br>Reset type: SYSRSn |
| 3 | OST4 | R | 0h | Latched Status Flag for OST4 Trip Latch<br>0: Reading a 0 indicates that no trip has occurred on OST4.<br>1: Reading a 1 indicates a trip has occured on the OST4 selected event.<br>Reset type: SYSRSn |
| 2 | OST3 | R | 0h | Latched Status Flag for OST3 Trip Latch<br>0: Reading a 0 indicates that no trip has occurred on OST3.<br>1: Reading a 1 indicates a trip has occured on the OST3 selected event.<br>Reset type: SYSRSn |
| 1 | OST2 | R | 0h | Latched Status Flag for OST2 Trip Latch<br>0: Reading a 0 indicates that no trip has occurred on OST2.<br>1: Reading a 1 indicates a trip has occured on the OST2 selected event.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-72. TZOSTFLG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | OST1 | R | 0h | Latched Status Flag for OST1 Trip Latch<br>0: Reading a 0 indicates that no trip has occurred on OST1.<br>1: Reading a 1 indicates a trip has occured on the OST1 selected event.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.52 TZCLR Register (Offset = 97h) [Reset = 0000h]

TZCLR is shown in Figure 15-145 and described in Table 15-73.

Return to the Summary Table.

Trip Zone Clear Register

**Figure 15-145. TZCLR Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| CBCPULSE | | RESERVED | | | | | |
| R/W-0h | | R-0-0h | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | DCBEVT2 | DCBEVT1 | DCAEVT2 | DCAEVT1 | OST | CBC | INT |
| R-0-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h |

**Table 15-73. TZCLR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-14 | CBCPULSE | R/W | 0h | Clear Pulse for Cycle-By-Cycle (CBC) Trip Latch<br>This bit field determines which pulse clears the CBC trip latch.<br>00: CTR = zero pulse clears CBC trip latch. (Same as legacy designs.)<br>01: CTR = PRD pulse clears CBC trip latch.<br>10: CTR = zero or CTR = PRD pulse clears CBC trip latch.<br>11: CBC trip latch is not cleared<br>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<br>0: Writing 0 has no effect. This bit always reads back 0.<br>1: Writing 1 clears the DCBEVT2 event trip condition.<br>Reset type: SYSRSn |
| 5 | DCBEVT1 | R-0/W1S | 0h | Clear Flag for Digital Compare Output B Event 1<br>0: Writing 0 has no effect. This bit always reads back 0.<br>1: Writing 1 clears the DCBEVT1 event trip condition.<br>Reset type: SYSRSn |
| 4 | DCAEVT2 | R-0/W1S | 0h | Clear Flag for Digital Compare Output A Event 2<br>0: Writing 0 has no effect. This bit always reads back 0.<br>1: Writing 1 clears the DCAEVT2 event trip condition.<br>Reset type: SYSRSn |
| 3 | DCAEVT1 | R-0/W1S | 0h | Clear Flag for Digital Compare Output A Event 1<br>0: Writing 0 has no effect. This bit always reads back 0.<br>1: Writing 1 clears the DCAEVT1 event trip condition.<br>Reset type: SYSRSn |
| 2 | OST | R-0/W1S | 0h | Clear Flag for One-Shot Trip (OST) Latch<br>0: Has no effect. Always reads back a 0.<br>1: Clears this Trip (set) condition.<br>Reset type: SYSRSn |
| 1 | CBC | R-0/W1S | 0h | Clear Flag for Cycle-By-Cycle (CBC) Trip Latch<br>0: Has no effect. Always reads back a 0.<br>1: Clears this Trip (set) condition.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-73. TZCLR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | INT | R-0/W1S | 0h | Global Interrupt Clear Flag<br>0: Has no effect. Always reads back a 0.<br>1: Clears the trip-interrupt flag for this ePWM module (TZFLG[INT]).<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.53 TZCBCCLR Register (Offset = 98h) [Reset = 0000h]

TZCBCCLR is shown in Figure 15-146 and described in Table 15-74.

Return to the Summary Table.

Trip Zone CBC Clear Register

**Figure 15-146. TZCBCCLR Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DCBEVT2 | DCAEVT2 | CBC6 | CBC5 | CBC4 | CBC3 | CBC2 | CBC1 |
| R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h |

**Table 15-74. 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<br>0: Writing a 0 has no effect.<br>1: Writing a 1 will clear the TZCBCFLG[DCBEVT2] bit.<br>Reset type: SYSRSn |
| 6 | DCAEVT2 | R-0/W1S | 0h | Clear Flag for Digital Compare Output A Event 2 selected for CBC<br>0: Writing a 0 has no effect.<br>1: Writing a 1 will clear the TZCBCFLG[DCAEVT2] bit.<br>Reset type: SYSRSn |
| 5 | CBC6 | R-0/W1S | 0h | Clear Flag for Cycle-By-Cycle (CBC6) Trip Latch<br>0: Writing a 0 has no effect.<br>1: Writing a 1 will clear the TZCBCFLG[CBC6] bit.<br>Reset type: SYSRSn |
| 4 | CBC5 | R-0/W1S | 0h | Clear Flag for Cycle-By-Cycle (CBC5) Trip Latch<br>0: Writing a 0 has no effect.<br>1: Writing a 1 will clear the TZCBCFLG[CBC5] bit.<br>Reset type: SYSRSn |
| 3 | CBC4 | R-0/W1S | 0h | Clear Flag for Cycle-By-Cycle (CBC4) Trip Latch<br>0: Writing a 0 has no effect.<br>1: Writing a 1 will clear the TZCBCFLG[CBC4] bit.<br>Reset type: SYSRSn |
| 2 | CBC3 | R-0/W1S | 0h | Clear Flag for Cycle-By-Cycle (CBC3) Trip Latch<br>0: Writing a 0 has no effect.<br>1: Writing a 1 will clear the TZCBCFLG[CBC3] bit.<br>Reset type: SYSRSn |
| 1 | CBC2 | R-0/W1S | 0h | Clear Flag for Cycle-By-Cycle (CBC2) Trip Latch<br>0: Writing a 0 has no effect.<br>1: Writing a 1 will clear the TZCBCFLG[CBC2] bit.<br>Reset type: SYSRSn |
| 0 | CBC1 | R-0/W1S | 0h | Clear Flag for Cycle-By-Cycle (CBC1) Trip Latch<br>0: Writing a 0 has no effect.<br>1: Writing a 1 will clear the TZCBCFLG[CBC1] bit.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.54 TZOSTCLR Register (Offset = 99h) [Reset = 0000h]

TZOSTCLR is shown in Figure 15-147 and described in Table 15-75.

Return to the Summary Table.

Trip Zone OST Clear Register

**Figure 15-147. TZOSTCLR Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DCBEVT1 | DCAEVT1 | OST6 | OST5 | OST4 | OST3 | OST2 | OST1 |
| R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h |

**Table 15-75. 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<br>0: Writing a 0 has no effect.<br>1: Writing a 1 will clear the TZOSTFLG[DCBEVT1] bit.<br>Reset type: SYSRSn |
| 6 | DCAEVT1 | R-0/W1S | 0h | Clear Flag for Digital Compare Output A Event 1 selected for OST<br>0: Writing a 0 has no effect.<br>1: Writing a 1 will clear the TZOSTFLG[DCAEVT1] bit.<br>Reset type: SYSRSn |
| 5 | OST6 | R-0/W1S | 0h | Clear Flag for Oneshot (OST6) Trip Latch<br>0: Writing a 0 has no effect.<br>1: Writing a 1 will clear the TZOSTFLG[OST6] bit.<br>Reset type: SYSRSn |
| 4 | OST5 | R-0/W1S | 0h | Clear Flag for Oneshot (OST5) Trip Latch<br>0: Writing a 0 has no effect.<br>1: Writing a 1 will clear the TZOSTFLG[OST5] bit.<br>Reset type: SYSRSn |
| 3 | OST4 | R-0/W1S | 0h | Clear Flag for Oneshot (OST4) Trip Latch<br>0: Writing a 0 has no effect.<br>1: Writing a 1 will clear the TZOSTFLG[OST4] bit.<br>Reset type: SYSRSn |
| 2 | OST3 | R-0/W1S | 0h | Clear Flag for Oneshot (OST3) Trip Latch<br>0: Writing a 0 has no effect.<br>1: Writing a 1 will clear the TZOSTFLG[OST3] bit.<br>Reset type: SYSRSn |
| 1 | OST2 | R-0/W1S | 0h | Clear Flag for Oneshot (OST2) Trip Latch<br>0: Writing a 0 has no effect.<br>1: Writing a 1 will clear the TZOSTFLG[OST2] bit.<br>Reset type: SYSRSn |
| 0 | OST1 | R-0/W1S | 0h | Clear Flag for Oneshot (OST1) Trip Latch<br>0: Writing a 0 has no effect.<br>1: Writing a 1 will clear the TZOSTFLG[OST1] bit.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.55 TZFRC Register (Offset = 9Bh) [Reset = 0000h]

TZFRC is shown in Figure 15-148 and described in Table 15-76.

Return to the Summary Table.

Trip Zone Force Register

**Figure 15-148. TZFRC Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | DCBEVT2 | DCBEVT1 | DCAEVT2 | DCAEVT1 | OST | CBC | RESERVED |
| R-0-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0-0h |

**Table 15-76. 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<br>0: Writing 0 has no effect. This bit always reads back 0.<br>1: Writing 1 forces the DCBEVT2 event trip condition and sets the TZFLG[DCBEVT2] bit.<br>Reset type: SYSRSn |
| 5 | DCBEVT1 | R-0/W1S | 0h | Force Flag for Digital Compare Output B Event 1<br>0: Writing 0 has no effect. This bit always reads back 0.<br>1: Writing 1 forces the DCBEVT1 event trip condition and sets the TZFLG[DCBEVT1] bit.<br>Reset type: SYSRSn |
| 4 | DCAEVT2 | R-0/W1S | 0h | Force Flag for Digital Compare Output A Event 2<br>0: Writing 0 has no effect. This bit always reads back 0.<br>1: Writing 1 forces the DCAEVT2 event trip condition and sets the TZFLG[DCAEVT2] bit.<br>Reset type: SYSRSn |
| 3 | DCAEVT1 | R-0/W1S | 0h | Force Flag for Digital Compare Output A Event 1<br>0: Writing 0 has no effect. This bit always reads back 0<br>1: Writing 1 forces the DCAEVT1 event trip condition and sets the TZFLG[DCAEVT1] bit.<br>Reset type: SYSRSn |
| 2 | OST | R-0/W1S | 0h | Force a One-Shot Trip Event via Software<br>0: Writing of 0 is ignored. Always reads back a 0.<br>1: Forces a one-shot trip event and sets the TZFLG[OST] bit.<br>Reset type: SYSRSn |
| 1 | CBC | R-0/W1S | 0h | Force a Cycle-by-Cycle Trip Event via Software<br>0: Writing of 0 is ignored. Always reads back a 0.<br>1: Forces a cycle-by-cycle trip event and sets the TZFLG[CBC] bit.<br>Reset type: SYSRSn |
| 0 | RESERVED | R-0 | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.56 ETSEL Register (Offset = A4h) [Reset = 0000h]

ETSEL is shown in Figure 15-149 and described in Table 15-77.

Return to the Summary Table.

Event Trigger Selection Register

**Figure 15-149. ETSEL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| SOCBEN | SOCBSEL | | | SOCAEN | SOCASEL | | |
| R/W-0h | R/W-0h | | | R/W-0h | R/W-0h | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | INTSELCMP | SOCBSELCMP | SOCASELCMP | INTEN | INTSEL | | |
| R-0-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | | |

**Table 15-77. ETSEL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | SOCBEN | R/W | 0h | Enable the ADC Start of Conversion B (EPWMxSOCB) Pulse<br>0: Disable EPWMxSOCB.<br>1: Enable EPWMxSOCB pulse.<br>Reset type: SYSRSn |
| 14-12 | SOCBSEL | R/W | 0h | EPWMxSOCB Selection Options<br>These bits determine when a EPWMxSOCB pulse will be generated.<br>000: Enable DCBEVT1.soc event<br>001: Enable event time-base counter equal to zero. (TBCTR = 0x00)<br>010: Enable event time-base counter equal to period (TBCTR = TBPRD)<br>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.<br>100: Enable event time-base counter equal to CMPA when the timer is incrementing or CMPC when the timer is incrementing<br>101: Enable event time-base counter equal to CMPA when the timer is decrementing or CMPC when the timer is decrementing<br>110: Enable event: time-base counter equal to CMPB when the timer is incrementing or CMPD when the timer is incrementing<br>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.<br>Reset type: SYSRSn |
| 11 | SOCAEN | R/W | 0h | Enable the ADC Start of Conversion A (EPWMxSOCA) Pulse<br>0: Disable EPWMxSOCA.<br>1: Enable EPWMxSOCA pulse.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-77. ETSEL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 10-8 | SOCASEL | R/W | 0h | EPWMxSOCA Selection Options<br>These bits determine when a EPWMxSOCA pulse will be generated.<br>000: Enable DCAEVT1.soc event<br>001: Enable event time-base counter equal to zero. (TBCTR = 0x00)<br>010: Enable event time-base counter equal to period (TBCTR = TBPRD)<br>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.<br>100: Enable event time-base counter equal to CMPA when the timer is incrementing or CMPC when the timer is incrementing<br>101: Enable event time-base counter equal to CMPA when the timer is decrementing or CMPC when the timer is decrementing<br>110: Enable event: time-base counter equal to CMPB when the timer is incrementing or CMPD when the timer is incrementing<br>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.<br>Reset type: SYSRSn |
| 7 | RESERVED | R-0 | 0h | Reserved |
| 6 | INTSELCMP | R/W | 0h | EPWMxINT Compare Register Selection Options<br>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.<br>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.<br>Reset type: SYSRSn |
| 5 | SOCBSELCMP | R/W | 0h | EPWMxSOCB Compare Register Selection Options<br>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.<br>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.<br>Reset type: SYSRSn |
| 4 | SOCASELCMP | R/W | 0h | EPWMxSOCA Compare Register Selection Options<br>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.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-77. ETSEL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3 | INTEN | R/W | 0h | Enable ePWM Interrupt (EPWMx_INT) Generation<br>0: Disable EPWMx_INT generation<br>1: Enable EPWMx_INT generation<br>Reset type: SYSRSn |
| 2-0 | INTSEL | R/W | 0h | ePWM Interrupt (EPWMx_INT) Selection Options<br>000: Reserved<br>001: Enable event time-base counter equal to zero. (TBCTR = 0x00)<br>010: Enable event time-base counter equal to period (TBCTR = TBPRD)<br>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.<br>100: Enable event time-base counter equal to CMPA when the timer is incrementing or CMPC when the timer is incrementing<br>101: Enable event time-base counter equal to CMPA when the timer is decrementing or CMPC when the timer is decrementing<br>110: Enable event: time-base counter equal to CMPB when the timer is incrementing or CMPD when the timer is incrementing<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.57 ETPS Register (Offset = A6h) [Reset = 0000h]

ETPS is shown in Figure 15-150 and described in Table 15-78.

Return to the Summary Table.

Event Trigger Pre-Scale Register

**Figure 15-150. ETPS Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| SOCBCNT | | SOCBPRD | | SOCACNT | | SOCAPRD | |
| R-0h | | R/W-0h | | R-0h | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | SOCPSSEL | INTPSSEL | INTCNT | | INTPRD | |
| R-0-0h | | R/W-0h | R/W-0h | R-0h | | R/W-0h | |

**Table 15-78. ETPS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-14 | SOCBCNT | R | 0h | ePWM ADC Start-of-Conversion B Event (EPWMxSOCB) Counter Register<br>These bits indicate how many selected ETSEL[SOCBSEL] events have occurred:<br>00: No events have occurred.<br>01: 1 event has occurred.<br>10: 2 events have occurred.<br>11: 3 events have occurred.<br>Reset type: SYSRSn |
| 13-12 | SOCBPRD | R/W | 0h | ePWM ADC Start-of-Conversion B Event (EPWMxSOCB) Period Select<br>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.<br>00: Disable the SOCB event counter. No EPWMxSOCB pulse will be generated<br>01: Generate the EPWMxSOCB pulse on the first event: ETPS[SOCBCNT] = 0,1<br>10: Generate the EPWMxSOCB pulse on the second event: ETPS[SOCBCNT] = 1,0<br>11: Generate the EPWMxSOCB pulse on the third event: ETPS[SOCBCNT] = 1,1<br>Reset type: SYSRSn |
| 11-10 | SOCACNT | R | 0h | ePWM ADC Start-of-Conversion A Event (EPWMxSOCA) Counter Register<br>These bits indicate how many selected ETSEL[SOCASEL] events have occurred:<br>00: No events have occurred.<br>01: 1 event has occurred.<br>10: 2 events have occurred.<br>11: 3 events have occurred.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-78. ETPS Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 9-8 | SOCAPRD | R/W | 0h | ePWM ADC Start-of-Conversion A Event (EPWMxSOCA) Period Select<br>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.<br>00: Disable the SOCA event counter. No EPWMxSOCA pulse will be generated<br>01: Generate the EPWMxSOCA pulse on the first event: ETPS[SOCACNT] = 0,1<br>10: Generate the EPWMxSOCA pulse on the second event: ETPS[SOCACNT] = 1,0<br>11: Generate the EPWMxSOCA pulse on the third event: ETPS[SOCACNT] = 1,1<br>Reset type: SYSRSn |
| 7-6 | RESERVED | R-0 | 0h | Reserved |
| 5 | SOCPSSEL | R/W | 0h | EPWMxSOC A/B Pre-Scale Selection Bits<br>0: Selects ETPS [SOCACNT/SOCBCNT] and [SOCAPRD/SOCBPRD] registers to determine frequency of events (SOC pulse once every 0-3 events).<br>1: Selects ETSOCPS [SOCACNT2/SOCBCNT2] and [SOCAPRD2/SOCBPRD2] registers to determine frequency of events (SOC pulse once every 0-15 events).<br>Reset type: SYSRSn |
| 4 | INTPSSEL | R/W | 0h | EPWMxINTn Pre-Scale Selection Bits<br>0: Selects ETPS [INTCNT, and INTPRD] registers to determine frequency of events (interrupt once every 0-3 events).<br>1: Selects ETINTPS [ INTCNT2, and INTPRD2 ] registers to determine frequency of events (interrupt once every 0-15 events).<br>Reset type: SYSRSn |
| 3-2 | INTCNT | R | 0h | ePWM Interrupt Event (EPWMx_INT) Counter Register<br>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].<br>00: No events have occurred.<br>01: 1 event has occurred.<br>10: 2 events have occurred.<br>11: 3 events have occurred.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-78. ETPS Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1-0 | INTPRD | R/W | 0h | ePWM Interrupt (EPWMx_INT) Period Select<br>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.<br>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.<br>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.<br>00: Disable the interrupt event counter. No interrupt will be generated and ETFRC[INT] is ignored.<br>01: Generate an interrupt on the first event INTCNT = 01 (first event)<br>10: Generate interrupt on ETPS[INTCNT] = 1,0 (second event)<br>11: Generate interrupt on ETPS[INTCNT] = 1,1 (third event)<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.58 ETFLG Register (Offset = A8h) [Reset = 0000h]

ETFLG is shown in Figure 15-151 and described in Table 15-79.

Return to the Summary Table.

Event Trigger Flag Register

**Figure 15-151. ETFLG Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | SOCB | SOCA | RESERVED | INT |
| R-0-0h | | | | R-0h | R-0h | R-0-0h | R-0h |

**Table 15-79. 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<br>Unlike the ETFLG[INT] flag, the EPWMxSOCB output will continue to pulse even if the flag bit is set.<br>0: Indicates no event occurred<br>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.<br>Reset type: SYSRSn |
| 2 | SOCA | R | 0h | Latched ePWM ADC Start-of-Conversion A (EPWMxSOCA) Status Flag<br>Unlike the ETFLG[INT] flag, the EPWMxSOCA output will continue to pulse even if the flag bit is set.<br>0: Indicates no event occurred<br>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.<br>Reset type: SYSRSn |
| 1 | RESERVED | R-0 | 0h | Reserved |
| 0 | INT | R | 0h | Latched ePWM Interrupt (EPWMx_INT) Status Flag<br>0: Indicates no event occurred<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.59 ETCLR Register (Offset = AAh) [Reset = 0000h]

ETCLR is shown in Figure 15-152 and described in Table 15-80.

Return to the Summary Table.

Event Trigger Clear Register

**Figure 15-152. ETCLR Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | SOCB | SOCA | RESERVED | INT |
| R-0-0h | | | | R-0/W1S-0h | R-0/W1S-0h | R-0-0h | R-0/W1S-0h |

**Table 15-80. 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<br>0: Writing a 0 has no effect. Always reads back a 0<br>1: Clears the ETFLG[SOCB] flag bit<br>Reset type: SYSRSn |
| 2 | SOCA | R-0/W1S | 0h | ePWM ADC Start-of-Conversion A (EPWMxSOCA) Flag Clear Bit<br>0: Writing a 0 has no effect. Always reads back a 0<br>1: Clears the ETFLG[SOCA] flag bit<br>Reset type: SYSRSn |
| 1 | RESERVED | R-0 | 0h | Reserved |
| 0 | INT | R-0/W1S | 0h | ePWM Interrupt (EPWMx_INT) Flag Clear Bit<br>0: Writing a 0 has no effect. Always reads back a 0<br>1: Clears the ETFLG[INT] flag bit and enable further interrupts pulses to be generated<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.60 ETFRC Register (Offset = ACh) [Reset = 0000h]

ETFRC is shown in Figure 15-153 and described in Table 15-81.

Return to the Summary Table.

Event Trigger Force Register

**Figure 15-153. ETFRC Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | SOCB | SOCA | RESERVED | INT |
| R-0-0h | | | | R-0/W1S-0h | R-0/W1S-0h | R-0-0h | R-0/W1S-0h |

**Table 15-81. 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<br>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.<br>0: Writing 0 to this bit will be ignored. Always reads back a 0.<br>1: Generates a pulse on EPWMxSOCB and set the SOCBFLG bit. This bit is used for test purposes.<br>Reset type: SYSRSn |
| 2 | SOCA | R-0/W1S | 0h | SOCA Force Bit<br>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.<br>0: Writing 0 to this bit will be ignored. Always reads back a 0.<br>1: Generates a pulse on EPWMxSOCA and set the SOCAFLG bit. This bit is used for test purposes.<br>Reset type: SYSRSn |
| 1 | RESERVED | R-0 | 0h | Reserved |
| 0 | INT | R-0/W1S | 0h | INT Force Bit<br>The interrupt will only be generated if the event is enabled in the ETSEL register. The INT flag bit will be set regardless.<br>0: Writing 0 to this bit will be ignored. Always reads back a 0.<br>1: Generates an interrupt on EPWMxINT and set the INT flag bit. This bit is used for test purposes.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.61 ETINTPS Register (Offset = AEh) [Reset = 0000h]

ETINTPS is shown in Figure 15-154 and described in Table 15-82.

Return to the Summary Table.

Event-Trigger Interrupt Pre-Scale Register

**Figure 15-154. ETINTPS Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTCNT2 | | | | INTPRD2 | | | |
| R-0h | | | | R/W-0h | | | |

**Table 15-82. ETINTPS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R-0 | 0h | Reserved |
| 7-4 | INTCNT2 | R | 0h | EPWMxINT Counter 2<br>When ETPS[INTPSSEL]=1, these bits indicate how many selected events have occurred:<br>0000: No events<br>0001: 1 event<br>0010: 2 events<br>0011: 3 events<br>0100: 4 events<br>...<br>1111: 15 events<br>Reset type: SYSRSn |
| 3-0 | INTPRD2 | R/W | 0h | EPWMxINT Period 2 Select<br>When ETPS[INTPSSEL] = 1, these bits select how many selected events need to occur before an interrupt is generated:<br>0000: Disable counter<br>0001: Generate interrupt on INTCNT = 1 (first event)<br>0010: Generate interrupt on INTCNT = 2 (second event)<br>0011: Generate interrupt on INTCNT = 3 (third event)<br>0100: Generate interrupt on INTCNT = 4 (fourth event)<br>...<br>1111: Generate interrupt on INTCNT = 15 (fifteenth event)<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.62 ETSOCPS Register (Offset = B0h) [Reset = 0000h]

ETSOCPS is shown in Figure 15-155 and described in Table 15-83.

Return to the Summary Table.

Event-Trigger SOC Pre-Scale Register

**Figure 15-155. ETSOCPS Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| SOCBCNT2 | | | | SOCBPRD2 | | | |
| R-0h | | | | R/W-0h | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SOCACNT2 | | | | SOCAPRD2 | | | |
| R-0h | | | | R/W-0h | | | |

**Table 15-83. ETSOCPS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | SOCBCNT2 | R | 0h | EPWMxSOCB Counter 2<br>When ETPS[SOCPSSEL] = 1, these bits indicate how many selected events have occurred:<br>0000: No events<br>0001: 1 event<br>0010: 2 events<br>0011: 3 events<br>0100: 4 events<br>...<br>1111: 15 events<br>Reset type: SYSRSn |
| 11-8 | SOCBPRD2 | R/W | 0h | EPWMxSOCB Period 2 Select<br>When ETPS[SOCPSSEL] = 1, these bits select how many selected event need to occur before an SOCB pulse is generated:<br>0000: Disable counter<br>0001: Generate SOC pulse on SOCBCNT2 = 1 (first event)<br>0010: Generate SOC pulse on SOCBCNT2 = 2 (second event)<br>0011: Generate SOC pulse on SOCBCNT2 = 3 (third event)<br>0100: Generate SOC pulse on SOCBCNT2 = 4 (fourth event)<br>...<br>1111: Generate SOC pulse on SOCBCNT2 = 15 (fifteenth event)<br>Reset type: SYSRSn |
| 7-4 | SOCACNT2 | R | 0h | EPWMxSOCA Counter 2<br>When ETPS[SOCPSSEL] = 1, these bits indicate how many selected events have occurred:<br>0000: No events<br>0001: 1 event<br>0010: 2 events<br>0011: 3 events<br>0100: 4 events<br>...<br>1111: 15 events<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-83. ETSOCPS Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-0 | SOCAPRD2 | R/W | 0h | EPWMxSOCA Period 2 Select<br>When ETPS[SOCPSSEL] = 1, these bits select how many selected event need to occur before an SOCA pulse is generated:<br>0000: Disable counter<br>0001: Generate SOC pulse on SOCACNT2 = 1 (first event)<br>0010: Generate SOC pulse on SOCACNT2 = 2 (second event)<br>0011: Generate SOC pulse on SOCACNT2 = 3 (third event)<br>0100: Generate SOC pulse on SOCACNT2 = 4 (fourth event)<br>...<br>1111: Generate SOC pulse on SOCACNT2 = 15 (fifteenth event)<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.63 ETCNTINITCTL Register (Offset = B2h) [Reset = 0000h]

ETCNTINITCTL is shown in Figure 15-156 and described in Table 15-84.

Return to the Summary Table.

Event-Trigger Counter Initialization Control Register

**Figure 15-156. ETCNTINITCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| SOCBINITEN | SOCAINITEN | INTINITEN | SOCBINITFRC | SOCAINITFRC | INTINITFRC | RESERVED | |
| R/W-0h | R/W-0h | R/W-0h | R-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 15-84. ETCNTINITCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | SOCBINITEN | R/W | 0h | EPWMxSOCB Counter 2 Initialization Enable<br>0: Has no effect.<br>1: Enable initialization of EPWMxSOCB counter with contents of ETCNTINIT[SOCBINIT] on a SYNC event or software force.<br>Reset type: SYSRSn |
| 14 | SOCAINITEN | R/W | 0h | EPWMxSOCA Counter 2 Initialization Enable<br>0: Has no effect.<br>1: Enable initialization of EPWMxSOCA counter with contents of ETCNTINIT[SOCAINIT] on a SYNC event or software force.<br>Reset type: SYSRSn |
| 13 | INTINITEN | R/W | 0h | EPWMxINT Counter 2 Initialization Enable<br>0: Has no effect.<br>1: Enable initialization of EPWMxINT counter 2 with contents of ETCNTINIT[INTINIT] on a SYNC event or software force.<br>Reset type: SYSRSn |
| 12 | SOCBINITFRC | R-0/W1S | 0h | EPWMxSOCB Counter 2 Initialization Force<br>0: Has no effect.<br>1: This bit forces the ET EPWMxSOCB counter to be initialized with the contents of ETCNTINIT[SOCBINIT].<br>Reset type: SYSRSn |
| 11 | SOCAINITFRC | R-0/W1S | 0h | EPWMxSOCA Counter 2 Initialization Force<br>0: Has no effect.<br>1: This bit forces the ET EPWMxSOCA counter to be initialized with the contents of ETCNTINIT[SOCAINIT].<br>Reset type: SYSRSn |
| 10 | INTINITFRC | R-0/W1S | 0h | EPWMxINT Counter 2 Initialization Force<br>0: Has no effect.<br>1: This bit forces the ET EPWMxINT counter to be initialized with the contents of ETCNTINIT[INTINIT].<br>Reset type: SYSRSn |
| 9-0 | RESERVED | R-0 | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.64 ETCNTINIT Register (Offset = B4h) [Reset = 0000h]

ETCNTINIT is shown in Figure 15-157 and described in Table 15-85.

Return to the Summary Table.

Event-Trigger Counter Initialization Register

**Figure 15-157. ETCNTINIT Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | SOCBINIT | | | |
| R-0h | | | | R/W-0h | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SOCAINIT | | | | INTINIT | | | |
| R/W-0h | | | | R/W-0h | | | |

**Table 15-85. 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<br>The ET EPWMxSOCB counter is initialized with the contents of this register on an ePWM SYNC event or a software force.<br>Reset type: SYSRSn |
| 7-4 | SOCAINIT | R/W | 0h | EPWMxSOCA Counter 2 Initialization Bits<br>The ET EPWMxSOCA counter is initialized with the contents of this register on an ePWM SYNC event or a software force.<br>Reset type: SYSRSn |
| 3-0 | INTINIT | R/W | 0h | EPWMxINT Counter 2 Initialization Bits<br>The ET EPWMxINT counter is initialized with the contents of this register on an ePWM SYNC event or a software force.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.65 DCTRIPSEL Register (Offset = C0h) [Reset = 0000h]

DCTRIPSEL is shown in Figure 15-158 and described in Table 15-86.

Return to the Summary Table.

Digital Compare Trip Select Register

**Figure 15-158. DCTRIPSEL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| DCBLCOMPSEL | | | | DCBHCOMPSEL | | | |
| R/W-0h | | | | R/W-0h | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DCALCOMPSEL | | | | DCAHCOMPSEL | | | |
| R/W-0h | | | | R/W-0h | | | |

**Table 15-86. DCTRIPSEL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | DCBLCOMPSEL | R/W | 0h | Digital Compare B Low Input Select Bits<br>0000: TRIPIN1<br>0001: TRIPIN2<br>0010: TRIPIN3<br>0011: TRIPIN4<br>...<br>1011: TRIPIN12<br>1100: Reserved<br>1101: TRIPIN14<br>1110: TRIPIN15<br>1111: Trip combination input (all trip inputs selected by DCBLTRIPSEL register ORed together)<br>Reset type: SYSRSn |
| 11-8 | DCBHCOMPSEL | R/W | 0h | Digital Compare B High Input Select Bits<br>0000: TRIPIN1<br>0001: TRIPIN2<br>0010: TRIPIN3<br>0011: TRIPIN4<br>...<br>1011: TRIPIN12<br>1100: Reserved<br>1101: TRIPIN14<br>1110: TRIPIN15<br>1111: Trip combination input (all trip inputs selected by DCBHTRIPSEL register ORed together)<br>Reset type: SYSRSn |
| 7-4 | DCALCOMPSEL | R/W | 0h | Digital Compare A Low Input Select Bits<br>0000: TRIPIN1<br>0001: TRIPIN2<br>0010: TRIPIN3<br>0011: TRIPIN4<br>...<br>1011: TRIPIN12<br>1100: Reserved<br>1101: TRIPIN14<br>1110: TRIPIN15<br>1111: Trip combination input (all trip inputs selected by DCALTRIPSEL register ORed together)<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-86. DCTRIPSEL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-0 | DCAHCOMPSEL | R/W | 0h | Digital Compare A High Input Select Bits<br>0000: TRIPIN1<br>0001: TRIPIN2<br>0010: TRIPIN3<br>0011: TRIPIN4<br>...<br>1011: TRIPIN12<br>1100: Reserved<br>1101: TRIPIN14<br>1110: TRIPIN15<br>1111: Trip combination input (all trip inputs selected by DCAHTRIPSEL register ORed together)<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.66 DCACTL Register (Offset = C3h) [Reset = 0000h]

DCACTL is shown in Figure 15-159 and described in Table 15-87.

Return to the Summary Table.

Digital Compare A Control Register

**Figure 15-159. DCACTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | EVT2FRCSYNCSEL | EVT2SRCSEL |
| R-0-0h | | | | | | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | EVT1SYNCE | EVT1SOCE | EVT1FRCSYNCSEL | EVT1SRCSEL |
| R-0-0h | | | | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 15-87. DCACTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-10 | RESERVED | R-0 | 0h | Reserved |
| 9 | EVT2FRCSYNCSEL | R/W | 0h | DCAEVT2 Force Synchronization Signal Select<br>0: Source is synchronized with EPWMCLK<br>1: Source is passed through asynchronously<br>Reset type: SYSRSn |
| 8 | EVT2SRCSEL | R/W | 0h | DCAEVT2 Source Signal Select<br>0: Source Is DCAEVT2 Signal<br>1: Source Is DCEVTFILT Signal<br>Reset type: SYSRSn |
| 7-4 | RESERVED | R-0 | 0h | Reserved |
| 3 | EVT1SYNCE | R/W | 0h | DCAEVT1 SYNC, Enable/Disable<br>0: SYNC Generation Disabled<br>1: SYNC Generation Enabled<br>Reset type: SYSRSn |
| 2 | EVT1SOCE | R/W | 0h | DCAEVT1 SOC, Enable/Disable<br>0: SOC Generation Disabled<br>1: SOC Generation Enabled<br>Reset type: SYSRSn |
| 1 | EVT1FRCSYNCSEL | R/W | 0h | DCAEVT1 Force Synchronization Signal Select<br>0: Source is synchronized with EPWMCLK<br>1: Source is passed through asynchronously<br>Reset type: SYSRSn |
| 0 | EVT1SRCSEL | R/W | 0h | DCAEVT1 Source Signal Select<br>0: Source Is DCAEVT1 Signal<br>1: Source Is DCEVTFILT Signal<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.67 DCBCTL Register (Offset = C4h) [Reset = 0000h]

DCBCTL is shown in Figure 15-160 and described in Table 15-88.

Return to the Summary Table.

Digital Compare B Control Register

**Figure 15-160. DCBCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | EVT2FRCSYNCSEL | EVT2SRCSEL |
| R-0-0h | | | | | | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | EVT1SYNCE | EVT1SOCE | EVT1FRCSYNCSEL | EVT1SRCSEL |
| R-0-0h | | | | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 15-88. DCBCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-10 | RESERVED | R-0 | 0h | Reserved |
| 9 | EVT2FRCSYNCSEL | R/W | 0h | DCBEVT2 Force Synchronization Signal Select<br>0: Source is synchronized with EPWMCLK<br>1: Source is passed through asynchronously<br>Reset type: SYSRSn |
| 8 | EVT2SRCSEL | R/W | 0h | DCBEVT2 Source Signal Select<br>0: Source Is DCBEVT2 Signal<br>1: Source Is DCEVTFILT Signal<br>Reset type: SYSRSn |
| 7-4 | RESERVED | R-0 | 0h | Reserved |
| 3 | EVT1SYNCE | R/W | 0h | DCBEVT1 SYNC, Enable/Disable<br>0: SYNC Generation Disabled<br>1: SYNC Generation Enabled<br>Reset type: SYSRSn |
| 2 | EVT1SOCE | R/W | 0h | DCBEVT1 SOC, Enable/Disable<br>0: SOC Generation Disabled<br>1: SOC Generation Enabled<br>Reset type: SYSRSn |
| 1 | EVT1FRCSYNCSEL | R/W | 0h | DCBEVT1 Force Synchronization Signal Select<br>0: Source is synchronized with EPWMCLK<br>1: Source is passed through asynchronously<br>Reset type: SYSRSn |
| 0 | EVT1SRCSEL | R/W | 0h | DCBEVT1 Source Signal Select<br>0: Source Is DCBEVT1 Signal<br>1: Source Is DCEVTFILT Signal<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.68 DCFCTL Register (Offset = C7h) [Reset = 0000h]

DCFCTL is shown in Figure 15-161 and described in Table 15-89.

Return to the Summary Table.

Digital Compare Filter Control Register

**Figure 15-161. DCFCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| EDGESTATUS | | | EDGECOUNT | | | EDGEMODE | |
| R-0h | | | R/W-0h | | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | EDGEFILTSEL | PULSESEL | | BLANKINV | BLANKE | SRCSEL | |
| R-0-0h | R/W-0h | R/W-0h | | R/W-0h | R/W-0h | R/W-0h | |

**Table 15-89. DCFCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-13 | EDGESTATUS | R | 0h | Edge Status:<br>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:<br>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:<br>000: no edges, reset current EDGESTATUS bits to 0,0,0<br>001: 1 edge<br>010: 2 edges<br>011: 3 edges<br>100: 4 edges<br>101: 5 edges<br>110: 6 edges<br>111: 7 edges<br>Reset type: SYSRSn |
| 9-8 | EDGEMODE | R/W | 0h | Edge Mode Select:<br>00: Low To High Edge<br>01: High To Low Edge<br>10: Both Edges<br>11: Reserved<br>Reset type: SYSRSn |
| 7 | RESERVED | R-0 | 0h | Reserved |
| 6 | EDGEFILTSEL | R/W | 0h | Edge Filter Select:<br>0: Edge Filter Not Selected<br>1: Edge Filter Selected<br>Reset type: SYSRSn |
| 5-4 | PULSESEL | R/W | 0h | Pulse Select For Blanking & Capture Alignment<br>00: Time-base counter equal to period (TBCTR = TBPRD)<br>01: Time-base counter equal to zero (TBCTR = 0x00)<br>10: Time-base counter equal to zero (TBCTR = 0x00) or period (TBCTR = TBPRD)<br>11: Reserved<br>Reset type: SYSRSn |
| 3 | BLANKINV | R/W | 0h | Blanking Window Inversion<br>0: Blanking window not inverted<br>1: Blanking window inverted<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-89. DCFCTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 2 | BLANKE | R/W | 0h | Blanking Window Enable/Disable<br>0: Blanking window is disabled<br>1: Blanking window is enabled<br>Reset type: SYSRSn |
| 1-0 | SRCSEL | R/W | 0h | Filter Block Signal Source Select<br>00: Source Is DCAEVT1 Signal<br>01: Source Is DCAEVT2 Signal<br>10: Source Is DCBEVT1 Signal<br>11: Source Is DCBEVT2 Signal<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.69 DCCAPCTL Register (Offset = C8h) [Reset = 0000h]

DCCAPCTL is shown in Figure 15-162 and described in Table 15-90.

Return to the Summary Table.

Digital Compare Capture Control Register

**Figure 15-162. DCCAPCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| CAPMODE | CAPCLR | CAPSTS | RESERVED | | | | |
| R/W-0h | R-0/W1S-0h | R-0h | R-0-0h | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | SHDWMODE | CAPE |
| R-0-0h | | | | | | R/W-0h | R/W-0h |

**Table 15-90. DCCAPCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | CAPMODE | R/W | 0h | Counter Capture Mode<br>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.<br>If active mode is enabled, via SHDWMODE bit in DCCAPCTL register, CPU reads of this register will return the active register value.<br>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.<br>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.<br>If active mode is enabled, via SHDWMODE bit in DCCAPCTL register, CPU reads of this register will return the active register value.<br>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.<br>Reset type: SYSRSn |
| 14 | CAPCLR | R-0/W1S | 0h | DC Capture Latched Status Clear Flag<br>0: Writing a 0 has no effect.<br>1: Writing a 1 will clear this CAPSTS (set) condition.<br>Reset type: SYSRSn |
| 13 | CAPSTS | R | 0h | Latched Status Flag for Capture Event<br>0: No DC capture event occurred.<br>1: A DC capture event has occurred.<br>Reset type: SYSRSn |
| 12-2 | RESERVED | R-0 | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-90. DCCAPCTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1 | SHDWMODE | R/W | 0h | TBCTR Counter Capture Shadow Select Mode<br>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.<br>1: Active Mode. In this mode the shadow register is disabled. CPU reads from the DCCAP register will always return the active register contents.<br>Reset type: SYSRSn |
| 0 | CAPE | R/W | 0h | TBCTR Counter Capture Enable/Disable<br>0: Disable the time-base counter capture.<br>1: Enable the time-base counter capture.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.70 DCFOFFSET Register (Offset = C9h) [Reset = 0000h]

DCFOFFSET is shown in Figure 15-163 and described in Table 15-91.

Return to the Summary Table.

Digital Compare Filter Offset Register

**Figure 15-163. DCFOFFSET Register**

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

**Table 15-91. DCFOFFSET Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | DCFOFFSET | R/W | 0h | Blanking Window Offset<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.71 DCFOFFSETCNT Register (Offset = CAh) [Reset = 0000h]

DCFOFFSETCNT is shown in Figure 15-164 and described in Table 15-92.

Return to the Summary Table.

Digital Compare Filter Offset Counter Register

**Figure 15-164. DCFOFFSETCNT Register**

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

**Table 15-92. DCFOFFSETCNT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | DCFOFFSETCNT | R | 0h | Blanking Offset Counter<br>These 16-bits are read only and indicate the current value of the offset counter. The counter counts down to zero and then stops until it is re-loaded on the next period or zero event as defined by the DCFCTL[PULSESEL] bit. The offset counter is not affected by the free/soft emulation bits. That is, it will always continue to count down if the device is halted by a emulation stop.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.72 DCFWINDOW Register (Offset = CBh) [Reset = 0000h]

DCFWINDOW is shown in Figure 15-165 and described in Table 15-93.

Return to the Summary Table.

Digital Compare Filter Window Register

**Figure 15-165. DCFWINDOW Register**

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

**Table 15-93. DCFWINDOW Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | DCFWINDOW | R/W | 0h | Blanking Window Width<br>00h: No blanking window is generated.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.73 DCFWINDOWCNT Register (Offset = CCh) [Reset = 0000h]

DCFWINDOWCNT is shown in Figure 15-166 and described in Table 15-94.

Return to the Summary Table.

Digital Compare Filter Window Counter Register

**Figure 15-166. DCFWINDOWCNT Register**

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

**Table 15-94. DCFWINDOWCNT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | DCFWINDOWCNT | R | 0h | Blanking Window Counter<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.74 DCCAP Register (Offset = CFh) [Reset = 0000h]

DCCAP is shown in Figure 15-167 and described in Table 15-95.

Return to the Summary Table.

Digital Compare Counter Capture Register

**Figure 15-167. DCCAP Register**

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

**Table 15-95. DCCAP Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | DCCAP | R | 0h | Digital Compare Time-Base Counter Capture<br>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.<br>- 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.<br>- 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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.75 DCAHTRIPSEL Register (Offset = D2h) [Reset = 0000h]

DCAHTRIPSEL is shown in Figure 15-168 and described in Table 15-96.

Return to the Summary Table.

Digital Compare AH Trip Select

**Figure 15-168. DCAHTRIPSEL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | TRIPINPUT15 | TRIPINPUT14 | RESERVED | TRIPINPUT12 | TRIPINPUT11 | TRIPINPUT10 | TRIPINPUT9 |
| R-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | 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 15-96. DCAHTRIPSEL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RESERVED | R | 0h | Reserved |
| 14 | TRIPINPUT15 | R/W | 0h | TRIP Input 15<br>0: Trip Input 15 not selected as combinational ORed input<br>1: Trip Input 15 selected as combinational ORed input to DCAH mux<br>Reset type: SYSRSn |
| 13 | TRIPINPUT14 | R/W | 0h | TRIP Input 14<br>0: Trip Input 14 not selected as combinational ORed input<br>1: Trip Input 14 selected as combinational ORed input to DCAH mux<br>Reset type: SYSRSn |
| 12 | RESERVED | R/W | 0h | Reserved |
| 11 | TRIPINPUT12 | R/W | 0h | TRIP Input 12<br>0: Trip Input 12 not selected as combinational ORed input<br>1: Trip Input 12 selected as combinational ORed input to DCAH mux<br>Reset type: SYSRSn |
| 10 | TRIPINPUT11 | R/W | 0h | TRIP Input 11<br>0: Trip Input 11 not selected as combinational ORed input<br>1: Trip Input 11 selected as combinational ORed input to DCAH mux<br>Reset type: SYSRSn |
| 9 | TRIPINPUT10 | R/W | 0h | TRIP Input 10<br>0: Trip Input 10 not selected as combinational ORed input<br>1: Trip Input 10 selected as combinational ORed input to DCAH mux<br>Reset type: SYSRSn |
| 8 | TRIPINPUT9 | R/W | 0h | TRIP Input 9<br>0: Trip Input 9 not selected as combinational ORed input<br>1: Trip Input 9 selected as combinational ORed input to DCAH mux<br>Reset type: SYSRSn |
| 7 | TRIPINPUT8 | R/W | 0h | TRIP Input 8<br>0: Trip Input 8 not selected as combinational ORed input<br>1: Trip Input 8 selected as combinational ORed input to DCAH mux<br>Reset type: SYSRSn |
| 6 | TRIPINPUT7 | R/W | 0h | TRIP Input 7<br>0: Trip Input 7 not selected as combinational ORed input<br>1: Trip Input 7 selected as combinational ORed input to DCAH mux<br>Reset type: SYSRSn |
| 5 | TRIPINPUT6 | R/W | 0h | TRIP Input 6<br>0: Trip Input 6 not selected as combinational ORed input<br>1: Trip Input 6 selected as combinational ORed input to DCAH mux<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-96. DCAHTRIPSEL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 4 | TRIPINPUT5 | R/W | 0h | TRIP Input 5<br>0: Trip Input 5 not selected as combinational ORed input<br>1: Trip Input 5 selected as combinational ORed input to DCAH mux<br>Reset type: SYSRSn |
| 3 | TRIPINPUT4 | R/W | 0h | TRIP Input 4<br>0: Trip Input 4 not selected as combinational ORed input<br>1: Trip Input 4 selected as combinational ORed input to DCAH mux<br>Reset type: SYSRSn |
| 2 | TRIPINPUT3 | R/W | 0h | TRIP Input 3<br>0: Trip Input 3 not selected as combinational ORed input<br>1: Trip Input 3 selected as combinational ORed input to DCAH mux<br>Reset type: SYSRSn |
| 1 | TRIPINPUT2 | R/W | 0h | TRIP Input 2<br>0: Trip Input 2 not selected as combinational ORed input<br>1: Trip Input 2 selected as combinational ORed input to DCAH mux<br>Reset type: SYSRSn |
| 0 | TRIPINPUT1 | R/W | 0h | TRIP Input 1<br>0: Trip Input 1 not selected as combinational ORed input<br>1: Trip Input 1 selected as combinational ORed input to DCAH mux<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.76 DCALTRIPSEL Register (Offset = D3h) [Reset = 0000h]

DCALTRIPSEL is shown in Figure 15-169 and described in Table 15-97.

Return to the Summary Table.

Digital Compare AL Trip Select

**Figure 15-169. DCALTRIPSEL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | TRIPINPUT15 | TRIPINPUT14 | RESERVED | TRIPINPUT12 | TRIPINPUT11 | TRIPINPUT10 | TRIPINPUT9 |
| R-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | 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 15-97. DCALTRIPSEL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RESERVED | R | 0h | Reserved |
| 14 | TRIPINPUT15 | R/W | 0h | TRIP Input 15<br>0: Trip Input 15 not selected as combinational ORed input<br>1: Trip Input 15 selected as combinational ORed input to DCAL mux<br>Reset type: SYSRSn |
| 13 | TRIPINPUT14 | R/W | 0h | TRIP Input 14<br>0: Trip Input 14 not selected as combinational ORed input<br>1: Trip Input 14 selected as combinational ORed input to DCAL mux<br>Reset type: SYSRSn |
| 12 | RESERVED | R/W | 0h | Reserved |
| 11 | TRIPINPUT12 | R/W | 0h | TRIP Input 12<br>0: Trip Input 12 not selected as combinational ORed input<br>1: Trip Input 12 selected as combinational ORed input to DCAL mux<br>Reset type: SYSRSn |
| 10 | TRIPINPUT11 | R/W | 0h | TRIP Input 11<br>0: Trip Input 11 not selected as combinational ORed input<br>1: Trip Input 11 selected as combinational ORed input to DCAL mux<br>Reset type: SYSRSn |
| 9 | TRIPINPUT10 | R/W | 0h | TRIP Input 10<br>0: Trip Input 10 not selected as combinational ORed input<br>1: Trip Input 10 selected as combinational ORed input to DCAL mux<br>Reset type: SYSRSn |
| 8 | TRIPINPUT9 | R/W | 0h | TRIP Input 9<br>0: Trip Input 9 not selected as combinational ORed input<br>1: Trip Input 9 selected as combinational ORed input to DCAL mux<br>Reset type: SYSRSn |
| 7 | TRIPINPUT8 | R/W | 0h | TRIP Input 8<br>0: Trip Input 8 not selected as combinational ORed input<br>1: Trip Input 8 selected as combinational ORed input to DCAL mux<br>Reset type: SYSRSn |
| 6 | TRIPINPUT7 | R/W | 0h | TRIP Input 7<br>0: Trip Input 7 not selected as combinational ORed input<br>1: Trip Input 7 selected as combinational ORed input to DCAL mux<br>Reset type: SYSRSn |
| 5 | TRIPINPUT6 | R/W | 0h | TRIP Input 6<br>0: Trip Input 6 not selected as combinational ORed input<br>1: Trip Input 6 selected as combinational ORed input to DCAL mux<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-97. DCALTRIPSEL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 4 | TRIPINPUT5 | R/W | 0h | TRIP Input 5<br>0: Trip Input 5 not selected as combinational ORed input<br>1: Trip Input 5 selected as combinational ORed input to DCAL mux<br>Reset type: SYSRSn |
| 3 | TRIPINPUT4 | R/W | 0h | TRIP Input 4<br>0: Trip Input 4 not selected as combinational ORed input<br>1: Trip Input 4 selected as combinational ORed input to DCAL mux<br>Reset type: SYSRSn |
| 2 | TRIPINPUT3 | R/W | 0h | TRIP Input 3<br>0: Trip Input 3 not selected as combinational ORed input<br>1: Trip Input 3 selected as combinational ORed input to DCAL mux<br>Reset type: SYSRSn |
| 1 | TRIPINPUT2 | R/W | 0h | TRIP Input 2<br>0: Trip Input 2 not selected as combinational ORed input<br>1: Trip Input 2 selected as combinational ORed input to DCAL mux<br>Reset type: SYSRSn |
| 0 | TRIPINPUT1 | R/W | 0h | TRIP Input 1<br>0: Trip Input 1 not selected as combinational ORed input<br>1: Trip Input 1 selected as combinational ORed input to DCAL mux<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.77 DCBHTRIPSEL Register (Offset = D4h) [Reset = 0000h]

DCBHTRIPSEL is shown in Figure 15-170 and described in Table 15-98.

Return to the Summary Table.

Digital Compare BH Trip Select

**Figure 15-170. DCBHTRIPSEL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | TRIPINPUT15 | TRIPINPUT14 | RESERVED | TRIPINPUT12 | TRIPINPUT11 | TRIPINPUT10 | TRIPINPUT9 |
| R-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | 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 15-98. DCBHTRIPSEL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RESERVED | R | 0h | Reserved |
| 14 | TRIPINPUT15 | R/W | 0h | TRIP Input 15<br>0: Trip Input 15 not selected as combinational ORed input<br>1: Trip Input 15 selected as combinational ORed input to DCBH mux<br>Reset type: SYSRSn |
| 13 | TRIPINPUT14 | R/W | 0h | TRIP Input 14<br>0: Trip Input 14 not selected as combinational ORed input<br>1: Trip Input 14 selected as combinational ORed input to DCBH mux<br>Reset type: SYSRSn |
| 12 | RESERVED | R/W | 0h | Reserved |
| 11 | TRIPINPUT12 | R/W | 0h | TRIP Input 12<br>0: Trip Input 12 not selected as combinational ORed input<br>1: Trip Input 12 selected as combinational ORed input to DCBH mux<br>Reset type: SYSRSn |
| 10 | TRIPINPUT11 | R/W | 0h | TRIP Input 11<br>0: Trip Input 11 not selected as combinational ORed input<br>1: Trip Input 11 selected as combinational ORed input to DCBH mux<br>Reset type: SYSRSn |
| 9 | TRIPINPUT10 | R/W | 0h | TRIP Input 10<br>0: Trip Input 10 not selected as combinational ORed input<br>1: Trip Input 10 selected as combinational ORed input to DCBH mux<br>Reset type: SYSRSn |
| 8 | TRIPINPUT9 | R/W | 0h | TRIP Input 9<br>0: Trip Input 9 not selected as combinational ORed input<br>1: Trip Input 9 selected as combinational ORed input to DCBH mux<br>Reset type: SYSRSn |
| 7 | TRIPINPUT8 | R/W | 0h | TRIP Input 8<br>0: Trip Input 8 not selected as combinational ORed input<br>1: Trip Input 8 selected as combinational ORed input to DCBH mux<br>Reset type: SYSRSn |
| 6 | TRIPINPUT7 | R/W | 0h | TRIP Input 7<br>0: Trip Input 7 not selected as combinational ORed input<br>1: Trip Input 7 selected as combinational ORed input to DCBH mux<br>Reset type: SYSRSn |
| 5 | TRIPINPUT6 | R/W | 0h | TRIP Input 6<br>0: Trip Input 6 not selected as combinational ORed input<br>1: Trip Input 6 selected as combinational ORed input to DCBH mux<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-98. DCBHTRIPSEL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 4 | TRIPINPUT5 | R/W | 0h | TRIP Input 5<br>0: Trip Input 5 not selected as combinational ORed input<br>1: Trip Input 5 selected as combinational ORed input to DCBH mux<br>Reset type: SYSRSn |
| 3 | TRIPINPUT4 | R/W | 0h | TRIP Input 4<br>0: Trip Input 4 not selected as combinational ORed input<br>1: Trip Input 4 selected as combinational ORed input to DCBH mux<br>Reset type: SYSRSn |
| 2 | TRIPINPUT3 | R/W | 0h | TRIP Input 3<br>0: Trip Input 3 not selected as combinational ORed input<br>1: Trip Input 3 selected as combinational ORed input to DCBH mux<br>Reset type: SYSRSn |
| 1 | TRIPINPUT2 | R/W | 0h | TRIP Input 2<br>0: Trip Input 2 not selected as combinational ORed input<br>1: Trip Input 2 selected as combinational ORed input to DCBH mux<br>Reset type: SYSRSn |
| 0 | TRIPINPUT1 | R/W | 0h | TRIP Input 1<br>0: Trip Input 1 not selected as combinational ORed input<br>1: Trip Input 1 selected as combinational ORed input to DCBH mux<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.78 DCBLTRIPSEL Register (Offset = D5h) [Reset = 0000h]

DCBLTRIPSEL is shown in Figure 15-171 and described in Table 15-99.

Return to the Summary Table.

Digital Compare BL Trip Select

**Figure 15-171. DCBLTRIPSEL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | TRIPINPUT15 | TRIPINPUT14 | RESERVED | TRIPINPUT12 | TRIPINPUT11 | TRIPINPUT10 | TRIPINPUT9 |
| R-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | 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 15-99. DCBLTRIPSEL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RESERVED | R | 0h | Reserved |
| 14 | TRIPINPUT15 | R/W | 0h | TRIP Input 15<br>0: Trip Input 15 not selected as combinational ORed input<br>1: Trip Input 15 selected as combinational ORed input to DCAL mux<br>Reset type: SYSRSn |
| 13 | TRIPINPUT14 | R/W | 0h | TRIP Input 14<br>0: Trip Input 14 not selected as combinational ORed input<br>1: Trip Input 14 selected as combinational ORed input to DCAL mux<br>Reset type: SYSRSn |
| 12 | RESERVED | R/W | 0h | Reserved |
| 11 | TRIPINPUT12 | R/W | 0h | TRIP Input 12<br>0: Trip Input 12 not selected as combinational ORed input<br>1: Trip Input 12 selected as combinational ORed input to DCAL mux<br>Reset type: SYSRSn |
| 10 | TRIPINPUT11 | R/W | 0h | TRIP Input 11<br>0: Trip Input 11 not selected as combinational ORed input<br>1: Trip Input 11 selected as combinational ORed input to DCAL mux<br>Reset type: SYSRSn |
| 9 | TRIPINPUT10 | R/W | 0h | TRIP Input 10<br>0: Trip Input 10 not selected as combinational ORed input<br>1: Trip Input 10 selected as combinational ORed input to DCAL mux<br>Reset type: SYSRSn |
| 8 | TRIPINPUT9 | R/W | 0h | TRIP Input 9<br>0: Trip Input 9 not selected as combinational ORed input<br>1: Trip Input 9 selected as combinational ORed input to DCAL mux<br>Reset type: SYSRSn |
| 7 | TRIPINPUT8 | R/W | 0h | TRIP Input 8<br>0: Trip Input 8 not selected as combinational ORed input<br>1: Trip Input 8 selected as combinational ORed input to DCAL mux<br>Reset type: SYSRSn |
| 6 | TRIPINPUT7 | R/W | 0h | TRIP Input 7<br>0: Trip Input 7 not selected as combinational ORed input<br>1: Trip Input 7 selected as combinational ORed input to DCAL mux<br>Reset type: SYSRSn |
| 5 | TRIPINPUT6 | R/W | 0h | TRIP Input 6<br>0: Trip Input 6 not selected as combinational ORed input<br>1: Trip Input 6 selected as combinational ORed input to DCAL mux<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-99. DCBLTRIPSEL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 4 | TRIPINPUT5 | R/W | 0h | TRIP Input 5<br>0: Trip Input 5 not selected as combinational ORed input<br>1: Trip Input 5 selected as combinational ORed input to DCAL mux<br>Reset type: SYSRSn |
| 3 | TRIPINPUT4 | R/W | 0h | TRIP Input 4<br>0: Trip Input 4 not selected as combinational ORed input<br>1: Trip Input 4 selected as combinational ORed input to DCAL mux<br>Reset type: SYSRSn |
| 2 | TRIPINPUT3 | R/W | 0h | TRIP Input 3<br>0: Trip Input 3 not selected as combinational ORed input<br>1: Trip Input 3 selected as combinational ORed input to DCAL mux<br>Reset type: SYSRSn |
| 1 | TRIPINPUT2 | R/W | 0h | TRIP Input 2<br>0: Trip Input 2 not selected as combinational ORed input<br>1: Trip Input 2 selected as combinational ORed input to DCAL mux<br>Reset type: SYSRSn |
| 0 | TRIPINPUT1 | R/W | 0h | TRIP Input 1<br>0: Trip Input 1 not selected as combinational ORed input<br>1: Trip Input 1 selected as combinational ORed input to DCAL mux<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.79 HWVDELVAL Register (Offset = FDh) [Reset = 0000h]

HWVDELVAL is shown in Figure 15-172 and described in Table 15-100.

Return to the Summary Table.

Hardware Valley Mode Delay Register

**Figure 15-172. 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 15-100. HWVDELVAL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | HWVDELVAL | R | 0h | Hardware Valley Delay Value Register<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.2.80 VCNTVAL Register (Offset = FEh) [Reset = 0000h]

VCNTVAL is shown in Figure 15-173 and described in Table 15-101.

Return to the Summary Table.

Hardware Valley Counter Register

**Figure 15-173. 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 15-101. VCNTVAL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | VCNTVAL | R | 0h | Valley Time Base Counter Register<br>This register reflects the captured VCNT value upon occurrence of STOPEDGE selected in VCNTCFG register.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 15.15.3 Register to Driverlib Function Mapping

#### 15.15.3.1 EPWM Registers to Driverlib Functions

**Table 15-102. EPWM Registers to Driverlib Functions**

| File | Driverlib Function |
|---|---|
| **TBCTL** | |
| epwm.c | EPWM_setEmulationMode |
| epwm.h | EPWM_setCountModeAfterSync |
| epwm.h | EPWM_setClockPrescaler |
| epwm.h | EPWM_forceSyncPulse |
| epwm.h | EPWM_setSyncOutPulseMode |
| epwm.h | EPWM_setPeriodLoadMode |
| epwm.h | EPWM_enablePhaseShiftLoad |
| epwm.h | EPWM_disablePhaseShiftLoad |
| epwm.h | EPWM_setTimeBaseCounterMode |
| epwm.h | EPWM_selectPeriodLoadEvent |
| epwm.h | EPWM_enableOneShotSync |
| epwm.h | EPWM_disableOneShotSync |
| epwm.h | EPWM_startOneShotSync |
| **TBCTL2** | |
| epwm.h | EPWM_setSyncOutPulseMode |
| epwm.h | EPWM_selectPeriodLoadEvent |
| epwm.h | EPWM_enableOneShotSync |
| epwm.h | EPWM_disableOneShotSync |
| epwm.h | EPWM_startOneShotSync |
| **TBCTR** | |
| epwm.h | EPWM_setTimeBaseCounter |
| epwm.h | EPWM_getTimeBaseCounterValue |
| **TBSTS** | |
| epwm.h | EPWM_getTimeBaseCounterOverflowStatus |
| epwm.h | EPWM_clearTimeBaseCounterOverflowEvent |
| epwm.h | EPWM_getSyncStatus |
| epwm.h | EPWM_clearSyncEvent |
| epwm.h | EPWM_getTimeBaseCounterDirection |
| **CMPCTL** | |
| epwm.h | EPWM_setCounterCompareShadowLoadMode |
| epwm.h | EPWM_disableCounterCompareShadowLoadMode |
| epwm.h | EPWM_getCounterCompareShadowStatus |
| **CMPCTL2** | |
| epwm.h | EPWM_setCounterCompareShadowLoadMode |
| epwm.h | EPWM_disableCounterCompareShadowLoadMode |
| **DBCTL** | |
| epwm.h | EPWM_setDeadBandOutputSwapMode |
| epwm.h | EPWM_setDeadBandDelayMode |
| epwm.h | EPWM_setDeadBandDelayPolarity |
| epwm.h | EPWM_setRisingEdgeDeadBandDelayInput |
| epwm.h | EPWM_setFallingEdgeDeadBandDelayInput |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-102. EPWM Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| 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 |
| **DBCTL2** | |
| epwm.h | EPWM_setDeadBandControlShadowLoadMode |
| epwm.h | EPWM_disableDeadBandControlShadowLoadMode |
| **AQCTL** | |
| epwm.h | EPWM_setActionQualifierShadowLoadMode |
| epwm.h | EPWM_disableActionQualifierShadowLoadMode |
| epwm.h | EPWM_setActionQualifierAction |
| epwm.h | EPWM_setActionQualifierActionComplete |
| epwm.h | EPWM_setAdditionalActionQualifierActionComplete |
| **AQTSRCSEL** | |
| epwm.h | EPWM_setActionQualifierT1TriggerSource |
| epwm.h | EPWM_setActionQualifierT2TriggerSource |
| **PCCTL** | |
| epwm.h | EPWM_enableChopper |
| epwm.h | EPWM_disableChopper |
| epwm.h | EPWM_setChopperDutyCycle |
| epwm.h | EPWM_setChopperFreq |
| epwm.h | EPWM_setChopperFirstPulseWidth |
| **VCAPCTL** | |
| 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 |
| **VCNTCFG** | |
| epwm.h | EPWM_setValleyTriggerEdgeCounts |
| epwm.h | EPWM_getValleyEdgeStatus |
| **HRCNFG** | |
| - | |
| **HRPWR** | |
| - | |
| **HRMSTEP** | |
| - | |
| **HRCNFG2** | |
| - | |
| **HRPCTL** | |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-102. EPWM Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| - | |
| **TRREM** | |
| - | |
| **GLDCTL** | |
| 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 |
| **GLDCFG** | |
| epwm.h | EPWM_enableGlobalLoadRegisters |
| epwm.h | EPWM_disableGlobalLoadRegisters |
| **XLINK** | |
| epwm.h | EPWM_setupEPWMLinks |
| **AQCTLA** | |
| epwm.h | EPWM_setActionQualifierAction |
| epwm.h | EPWM_setActionQualifierActionComplete |
| epwm.h | EPWM_setAdditionalActionQualifierActionComplete |
| **AQCTLA2** | |
| epwm.h | EPWM_setActionQualifierAction |
| epwm.h | EPWM_setAdditionalActionQualifierActionComplete |
| **AQCTLB** | |
| - | See AQCTLA |
| **AQCTLB2** | |
| - | See AQCTLA2 |
| **AQSFRC** | |
| epwm.h | EPWM_setActionQualifierContSWForceShadowMode |
| epwm.h | EPWM_setActionQualifierSWAction |
| epwm.h | EPWM_forceActionQualifierSWAction |
| **AQCSFRC** | |
| epwm.h | EPWM_setActionQualifierContSWForceAction |
| **DBREDHR** | |
| - | |
| **DBRED** | |
| epwm.h | EPWM_setRisingEdgeDelayCount |
| **DBFEDHR** | |
| - | |
| **DBFED** | |
| epwm.h | EPWM_setFallingEdgeDelayCount |
| **TBPHS** | |
| epwm.h | EPWM_setPhaseShift |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-102. EPWM Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| **TBPRDHR** | |
| - | |
| **TBPRD** | |
| epwm.h | EPWM_setTimeBasePeriod |
| epwm.h | EPWM_getTimeBasePeriod |
| **CMPA** | |
| epwm.h | EPWM_setCounterCompareValue |
| epwm.h | EPWM_getCounterCompareValue |
| **CMPB** | |
| - | See CMPA |
| **CMPC** | |
| epwm.h | EPWM_setCounterCompareShadowLoadMode |
| epwm.h | EPWM_disableCounterCompareShadowLoadMode |
| epwm.h | EPWM_getCounterCompareShadowStatus |
| **CMPD** | |
| - | See CMPC |
| **GLDCTL2** | |
| epwm.h | EPWM_setGlobalLoadOneShotLatch |
| epwm.h | EPWM_forceGlobalLoadOneShotEvent |
| **SWVDELVAL** | |
| epwm.h | EPWM_setValleySWDelayValue |
| **TZSEL** | |
| epwm.h | EPWM_enableTripZoneSignals |
| epwm.h | EPWM_disableTripZoneSignals |
| **TZDCSEL** | |
| epwm.h | EPWM_setTripZoneDigitalCompareEventCondition |
| **TZCTL** | |
| epwm.h | EPWM_enableTripZoneAdvAction |
| epwm.h | EPWM_disableTripZoneAdvAction |
| epwm.h | EPWM_setTripZoneAction |
| epwm.h | EPWM_setTripZoneAdvAction |
| epwm.h | EPWM_setTripZoneAdvDigitalCompareActionA |
| epwm.h | EPWM_setTripZoneAdvDigitalCompareActionB |
| **TZCTL2** | |
| epwm.h | EPWM_enableTripZoneAdvAction |
| epwm.h | EPWM_disableTripZoneAdvAction |
| epwm.h | EPWM_setTripZoneAdvAction |
| epwm.h | EPWM_setTripZoneAdvDigitalCompareActionA |
| epwm.h | EPWM_setTripZoneAdvDigitalCompareActionB |
| **TZCTLDCA** | |
| epwm.h | EPWM_setTripZoneAdvDigitalCompareActionA |
| **TZCTLDCB** | |
| epwm.h | EPWM_setTripZoneAdvDigitalCompareActionB |
| **TZEINT** | |
| epwm.h | EPWM_enableTripZoneInterrupt |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-102. EPWM Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| epwm.h | EPWM_disableTripZoneInterrupt |
| **TZFLG** | |
| epwm.h | EPWM_getTripZoneFlagStatus |
| **TZCBCFLG** | |
| epwm.h | EPWM_getCycleByCycleTripZoneFlagStatus |
| **TZOSTFLG** | |
| epwm.h | EPWM_getOneShotTripZoneFlagStatus |
| **TZCLR** | |
| epwm.h | EPWM_selectCycleByCycleTripZoneClearEvent |
| epwm.h | EPWM_clearTripZoneFlag |
| **TZCBCCLR** | |
| epwm.h | EPWM_clearCycleByCycleTripZoneFlag |
| **TZOSTCLR** | |
| epwm.h | EPWM_clearOneShotTripZoneFlag |
| **TZFRC** | |
| epwm.h | EPWM_forceTripZoneEvent |
| **ETSEL** | |
| epwm.h | EPWM_enableInterrupt |
| epwm.h | EPWM_disableInterrupt |
| epwm.h | EPWM_setInterruptSource |
| epwm.h | EPWM_enableADCTrigger |
| epwm.h | EPWM_disableADCTrigger |
| epwm.h | EPWM_setADCTriggerSource |
| **ETPS** | |
| epwm.h | EPWM_setInterruptEventCount |
| epwm.h | EPWM_setADCTriggerEventPrescale |
| **ETFLG** | |
| epwm.h | EPWM_getEventTriggerInterruptStatus |
| epwm.h | EPWM_getADCTriggerFlagStatus |
| **ETCLR** | |
| epwm.h | EPWM_clearEventTriggerInterruptFlag |
| epwm.h | EPWM_clearADCTriggerFlag |
| **ETFRC** | |
| epwm.h | EPWM_forceEventTriggerInterrupt |
| epwm.h | EPWM_forceADCTrigger |
| **ETINTPS** | |
| epwm.h | EPWM_setInterruptEventCount |
| epwm.h | EPWM_getInterruptEventCount |
| **ETSOCPS** | |
| epwm.h | EPWM_setADCTriggerEventPrescale |
| epwm.h | EPWM_getADCTriggerEventCount |
| **ETCNTINITCTL** | |
| epwm.h | EPWM_enableInterruptEventCountInit |
| epwm.h | EPWM_disableInterruptEventCountInit |
| epwm.h | EPWM_forceInterruptEventCountInit |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-102. EPWM Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| epwm.h | EPWM_enableADCTriggerEventCountInit |
| epwm.h | EPWM_disableADCTriggerEventCountInit |
| epwm.h | EPWM_forceADCTriggerEventCountInit |
| **ETCNTINIT** | |
| 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 |
| **DCTRIPSEL** | |
| epwm.h | EPWM_selectDigitalCompareTripInput |
| epwm.h | EPWM_enableDigitalCompareTripCombinationInput |
| **DCACTL** | |
| epwm.h | EPWM_setDigitalCompareEventSource |
| epwm.h | EPWM_setDigitalCompareEventSyncMode |
| epwm.h | EPWM_enableDigitalCompareADCTrigger |
| epwm.h | EPWM_disableDigitalCompareADCTrigger |
| epwm.h | EPWM_enableDigitalCompareSyncEvent |
| epwm.h | EPWM_disableDigitalCompareSyncEvent |
| **DCBCTL** | |
| - | See DCACTL |
| **DCFCTL** | |
| 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 |
| epwm.h | EPWM_getDigitalCompareEdgeFilterEdgeStatus |
| **DCCAPCTL** | |
| epwm.h | EPWM_enableDigitalCompareCounterCapture |
| epwm.h | EPWM_disableDigitalCompareCounterCapture |
| epwm.h | EPWM_setDigitalCompareCounterShadowMode |
| epwm.h | EPWM_getDigitalCompareCaptureStatus |
| epwm.h | EPWM_clearDigitalCompareCaptureStatusFlag |
| epwm.h | EPWM_configureDigitalCompareCounterCaptureMode |
| **DCFOFFSET** | |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-102. EPWM Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| epwm.h | EPWM_setDigitalCompareWindowOffset |
| epwm.h | EPWM_getDigitalCompareBlankingWindowOffsetCount |
| **DCFOFFSETCNT** | |
| epwm.h | EPWM_getDigitalCompareBlankingWindowOffsetCount |
| **DCFWINDOW** | |
| epwm.h | EPWM_setDigitalCompareWindowLength |
| epwm.h | EPWM_getDigitalCompareBlankingWindowLengthCount |
| **DCFWINDOWCNT** | |
| epwm.h | EPWM_getDigitalCompareBlankingWindowLengthCount |
| **DCCAP** | |
| 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 |
| **DCAHTRIPSEL** | |
| epwm.h | EPWM_enableDigitalCompareTripCombinationInput |
| epwm.h | EPWM_disableDigitalCompareTripCombinationInput |
| **DCALTRIPSEL** | |
| - | See DCAHTRIPSEL |
| **DCBHTRIPSEL** | |
| - | See DCAHTRIPSEL |
| **DCBLTRIPSEL** | |
| - | See DCAHTRIPSEL |
| **HWVDELVAL** | |
| epwm.h | EPWM_getValleyHWDelay |
| **VCNTVAL** | |
| epwm.h | EPWM_getValleyCount |

### 15.15.3.2 HRPWM Registers to Driverlib Functions

**Table 15-103. HRPWM Registers to Driverlib Functions**

| File | Driverlib Function |
|---|---|
| **TBCTL** | |
| - | |
| **TBCTL2** | |
| - | |
| **TBCTR** | |
| - | |
| **TBSTS** | |
| - | |
| **CMPCTL** | |
| - | |
| **CMPCTL2** | |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-103. HRPWM Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| - | |
| **DBCTL** | |
| - | |
| **DBCTL2** | |
| - | |
| **AQCTL** | |
| - | |
| **AQTSRCSEL** | |
| - | |
| **PCCTL** | |
| - | |
| **VCAPCTL** | |
| - | |
| **VCNTCFG** | |
| - | |
| **HRCNFG** | |
| 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 |
| **HRPWR** | |
| - | |
| **HRMSTEP** | |
| hrpwm.h | HRPWM_setMEPStep |
| **HRCNFG2** | |
| hrpwm.h | HRPWM_setDeadbandMEPEdgeSelect |
| hrpwm.h | HRPWM_setRisingEdgeDelayLoadMode |
| hrpwm.h | HRPWM_setFallingEdgeDelayLoadMode |
| **HRPCTL** | |
| hrpwm.h | HRPWM_enablePeriodControl |
| hrpwm.h | HRPWM_disablePeriodControl |
| hrpwm.h | HRPWM_enablePhaseShiftLoad |
| hrpwm.h | HRPWM_disablePhaseShiftLoad |
| hrpwm.h | HRPWM_setSyncPulseSource |
| **TRREM** | |
| hrpwm.h | HRPWM_setTranslatorRemainder |
| **GLDCTL** | |
| - | |
| **GLDCFG** | |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-103. HRPWM Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| - | |
| **EPWMXLINK** | |
| - | |
| **AQCTLA** | |
| - | |
| **AQCTLA2** | |
| - | |
| **AQCTLB** | |
| - | |
| **AQCTLB2** | |
| - | |
| **AQSFRC** | |
| - | |
| **AQCSFRC** | |
| - | |
| **DBREDHR** | |
| hrpwm.h | HRPWM_setRisingEdgeDelay |
| hrpwm.h | HRPWM_setHiResRisingEdgeDelayOnly |
| **DBRED** | |
| hrpwm.h | HRPWM_setRisingEdgeDelay |
| hrpwm.h | HRPWM_setHiResRisingEdgeDelayOnly |
| **DBFEDHR** | |
| hrpwm.h | HRPWM_setFallingEdgeDelay |
| hrpwm.h | HRPWM_setHiResFallingEdgeDelayOnly |
| **DBFED** | |
| hrpwm.h | HRPWM_setFallingEdgeDelay |
| hrpwm.h | HRPWM_setHiResFallingEdgeDelayOnly |
| **TBPHS** | |
| hrpwm.h | HRPWM_setPhaseShift |
| hrpwm.h | HRPWM_setHiResPhaseShiftOnly |
| **TBPRDHR** | |
| hrpwm.h | HRPWM_setTimeBasePeriod |
| hrpwm.h | HRPWM_setHiResTimeBasePeriodOnly |
| hrpwm.h | HRPWM_getTimeBasePeriod |
| hrpwm.h | HRPWM_getHiResTimeBasePeriodOnly |
| **TBPRD** | |
| hrpwm.h | HRPWM_setTimeBasePeriod |
| hrpwm.h | HRPWM_setHiResTimeBasePeriodOnly |
| hrpwm.h | HRPWM_getTimeBasePeriod |
| hrpwm.h | HRPWM_getHiResTimeBasePeriodOnly |
| **CMPA** | |
| hrpwm.h | HRPWM_setCounterCompareValue |
| hrpwm.h | HRPWM_setHiResCounterCompareValueOnly |
| hrpwm.h | HRPWM_getCounterCompareValue |
| hrpwm.h | HRPWM_getHiResCounterCompareValueOnly |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-103. HRPWM Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| **CMPB** | |
| hrpwm.h | HRPWM_setCounterCompareValue |
| hrpwm.h | HRPWM_setHiResCounterCompareValueOnly |
| hrpwm.h | HRPWM_getCounterCompareValue |
| hrpwm.h | HRPWM_getHiResCounterCompareValueOnly |
| **CMPC** | |
| - | |
| **CMPD** | |
| - | |
| **GLDCTL2** | |
| - | |
| **SWVDELVAL** | |
| - | |
| **TZSEL** | |
| - | |
| **TZDCSEL** | |
| - | |
| **TZCTL** | |
| - | |
| **TZCTL2** | |
| - | |
| **TZCTLDCA** | |
| - | |
| **TZCTLDCB** | |
| - | |
| **TZEINT** | |
| - | |
| **TZFLG** | |
| - | |
| **TZCBCFLG** | |
| - | |
| **TZOSTFLG** | |
| - | |
| **TZCLR** | |
| - | |
| **TZCBCCLR** | |
| - | |
| **TZOSTCLR** | |
| - | |
| **TZFRC** | |
| - | |
| **ETSEL** | |
| - | |
| **ETPS** | |
| - | |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-103. HRPWM Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| **ETFLG** | |
| - | |
| **ETCLR** | |
| - | |
| **ETFRC** | |
| - | |
| **ETINTPS** | |
| - | |
| **ETSOCPS** | |
| - | |
| **ETCNTINITCTL** | |
| - | |
| **ETCNTINIT** | |
| - | |
| **DCTRIPSEL** | |
| - | |
| **DCACTL** | |
| - | |
| **DCBCTL** | |
| - | |
| **DCFCTL** | |
| - | |
| **DCCAPCTL** | |
| - | |
| **DCFOFFSET** | |
| - | |
| **DCFOFFSETCNT** | |
| - | |
| **DCFWINDOW** | |
| - | |
| **DCFWINDOWCNT** | |
| - | |
| **DCCAP** | |
| - | |
| **DCAHTRIPSEL** | |
| - | |
| **DCALTRIPSEL** | |
| - | |
| **DCBHTRIPSEL** | |
| - | |
| **DCBLTRIPSEL** | |
| - | |
| **HWVDELVAL** | |
| - | |
| **VCNTVAL** | |



Copyright © 2024 Texas Instruments Incorporated

**Table 15-103. HRPWM Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| - | |



Copyright © 2024 Texas Instruments Incorporated

Chapter 16

# 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 0 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.


**16.1 Introduction** ..... 2154
**16.2 Description** ..... 2154
**16.3 Configuring Device Pins for the eCAP** ..... 2155
**16.4 Capture and APWM Operating Mode** ..... 2156
**16.5 Capture Mode Description** ..... 2158
**16.6 Application of the eCAP Module** ..... 2167
**16.7 Application of the APWM Mode** ..... 2171
**16.8 Software** ..... 2172
**16.9 eCAP Registers** ..... 2173





Copyright © 2024 Texas Instruments Incorporated

## 16.1 Introduction

### 16.1.1 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
- All above resources are dedicated to a single input pin
- When not used in capture mode, the eCAP module can be configured as a single-channel PWM output

### 16.1.2 ECAP Related Collateral

**Foundational Materials**

- C2000 Academy - ECAP

**Getting Started Materials**

- *Leveraging High Resolution Capture (HRCAP) for Single Wire Data Transfer Application Report*

## 16.2 Description

The eCAP module represents one complete capture channel that can be instantiated multiple times, depending on the target device. In the context of this guide, one eCAP channel has the following independent key resources:

- Dedicated input capture pin
- 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.
- 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
- Interrupt capabilities on any of the four capture events



Copyright © 2024 Texas Instruments Incorporated

## 16.3 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 ±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.

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.

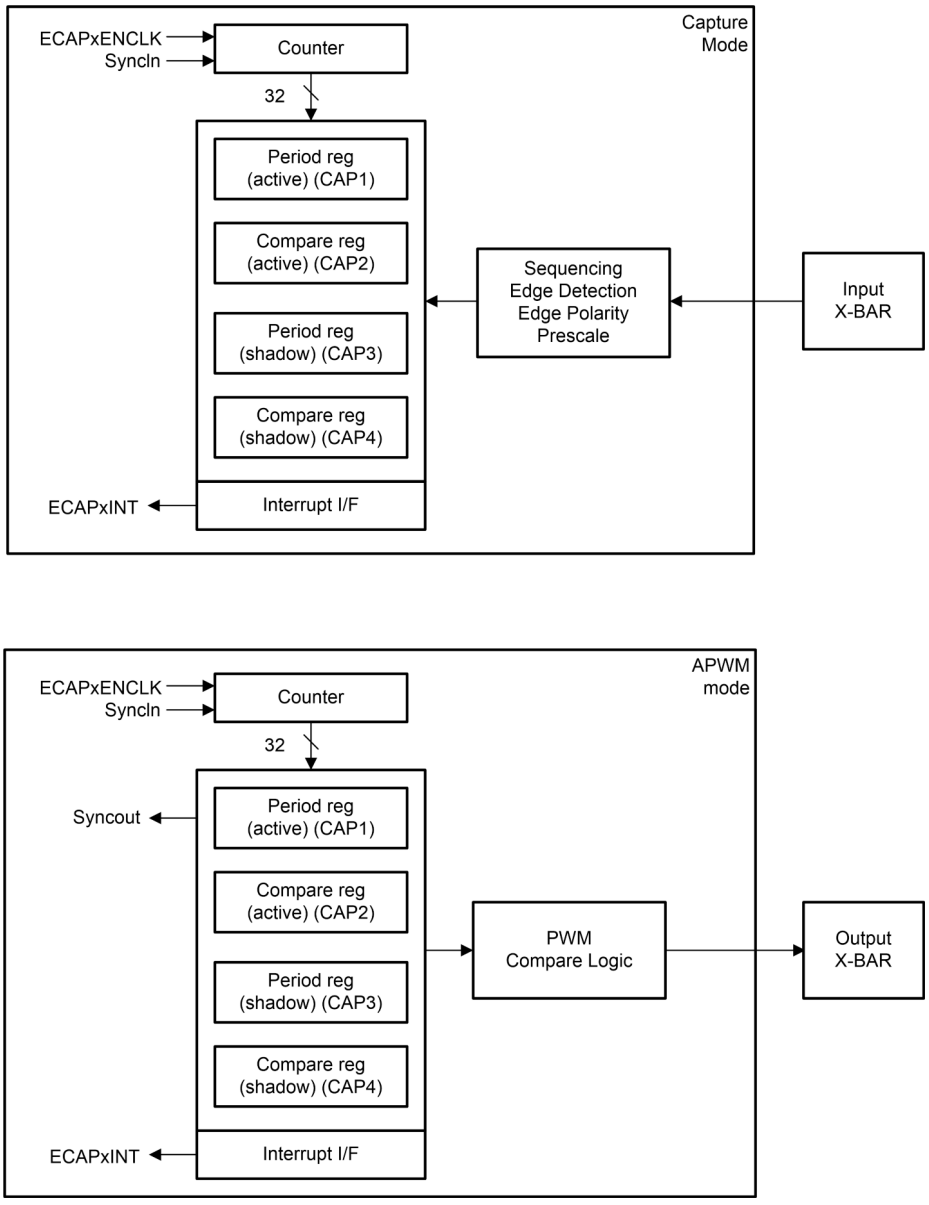


Copyright © 2024 Texas Instruments Incorporated

## 16.4 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 16-1 is a high-level view of both the capture and auxiliary pulse-width modulator (APWM) modes of operation.

Figure 16-2 further descries the output of the eCAP in APWM mode based on the CMP and PRD values.




A. 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.

B. In APWM mode, writing any value to CAP1/CAP2 active registers also writes the same value to the corresponding shadow registers CAP3/CAP4. This emulates immediate mode. Writing to the shadow registers CAP3/CAP4 invokes the shadow mode.

**Figure 16-1. Capture and APWM Modes of Operation**



Copyright © 2024 Texas Instruments Incorporated

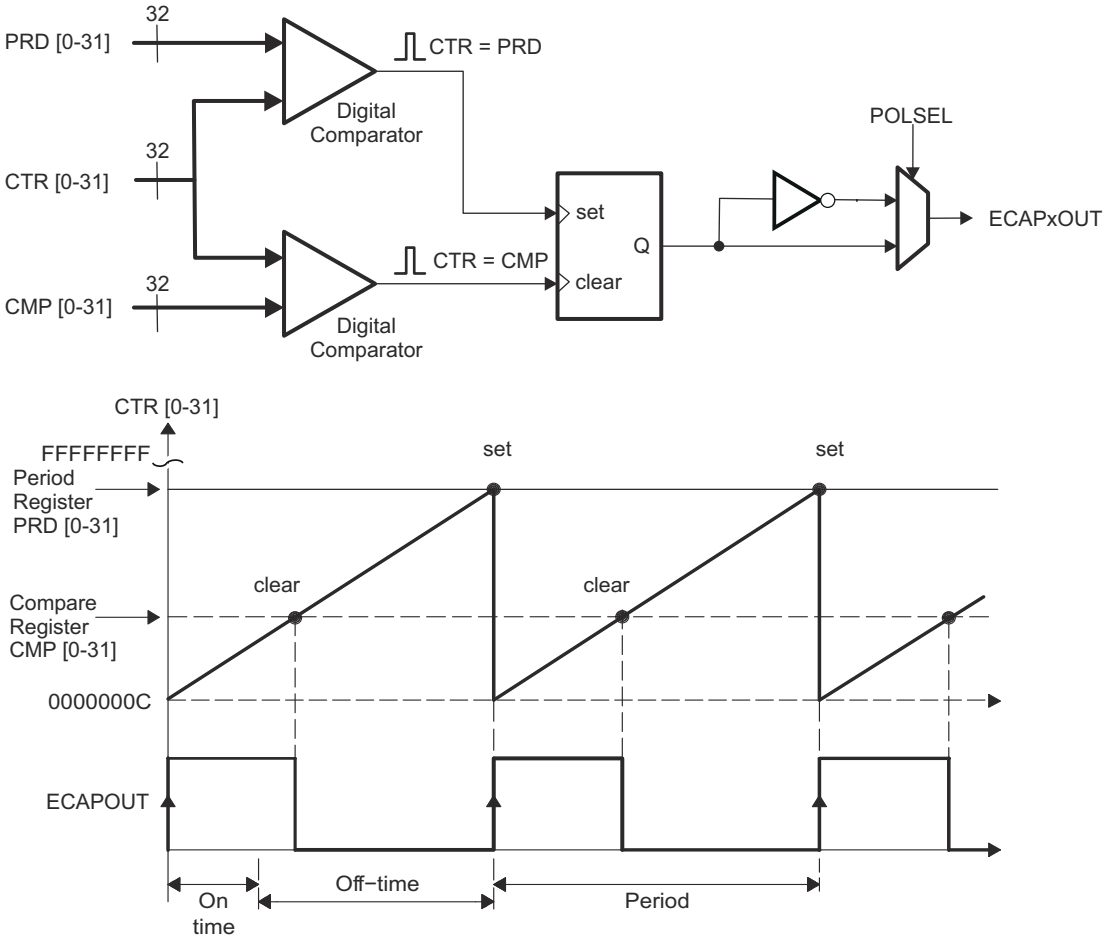



**Figure 16-2. Counter Compare and PRD Effects on the eCAP Output in APWM Mode**



Copyright © 2024 Texas Instruments Incorporated

## 16.5 Capture Mode Description

Figure 16-3 shows the various components that implement the capture function.

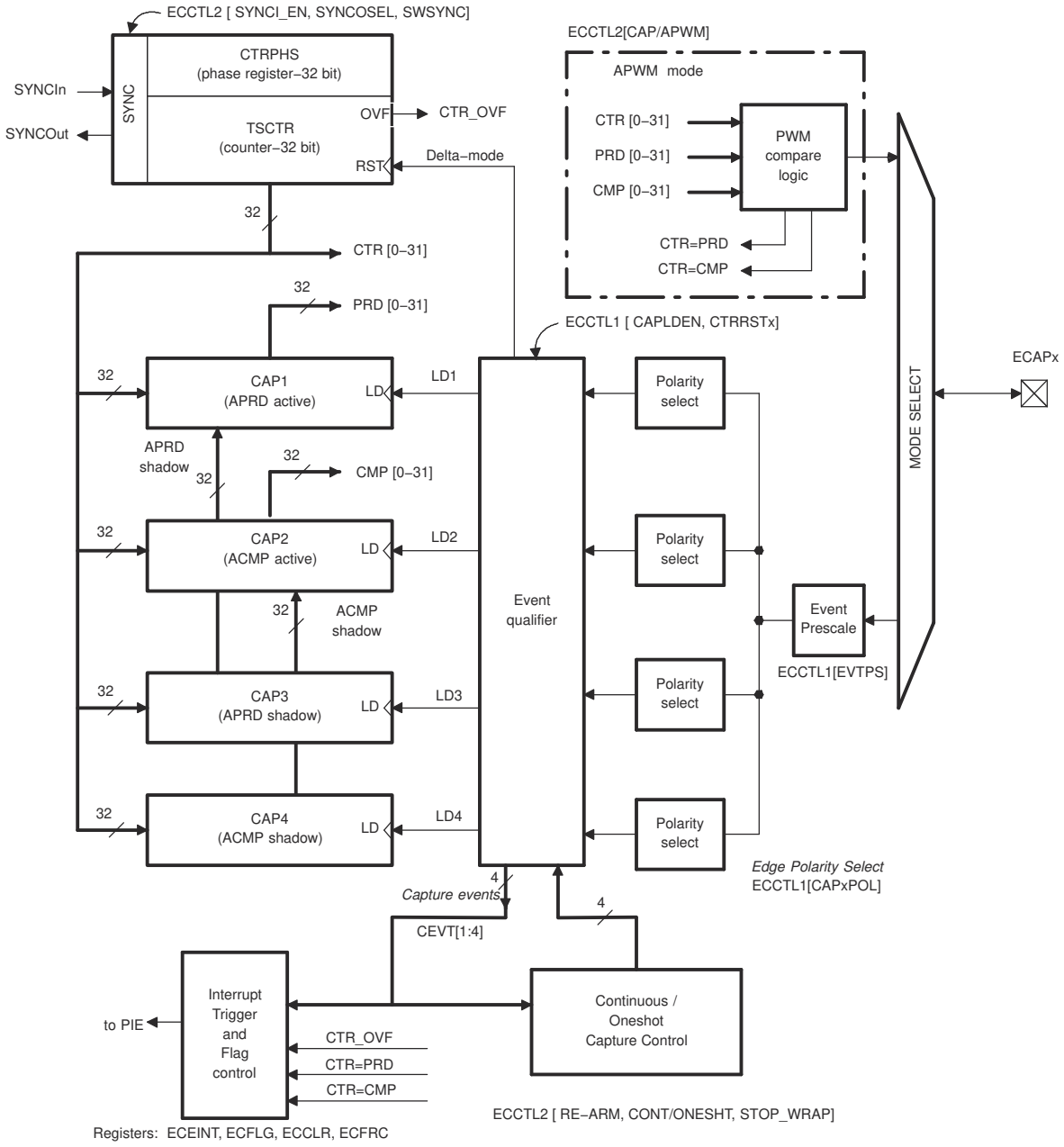


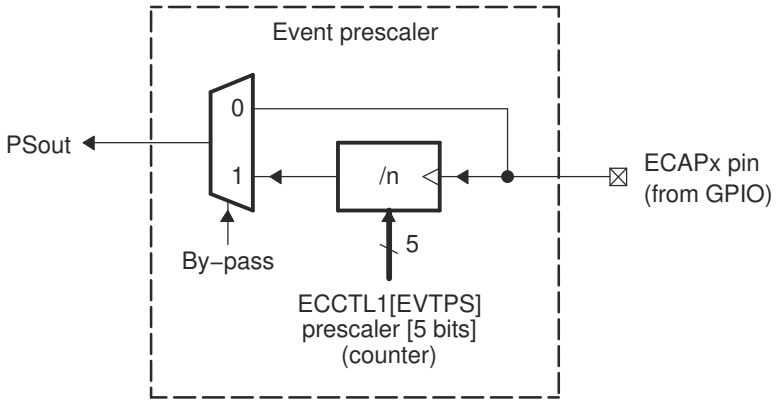

**Figure 16-3. eCAP Block Diagram**



Copyright © 2024 Texas Instruments Incorporated

### 16.5.1 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 16-4 shows a functional diagram and Figure 16-5 shows the operation of the prescale function.




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 16-4. Event Prescale Control**

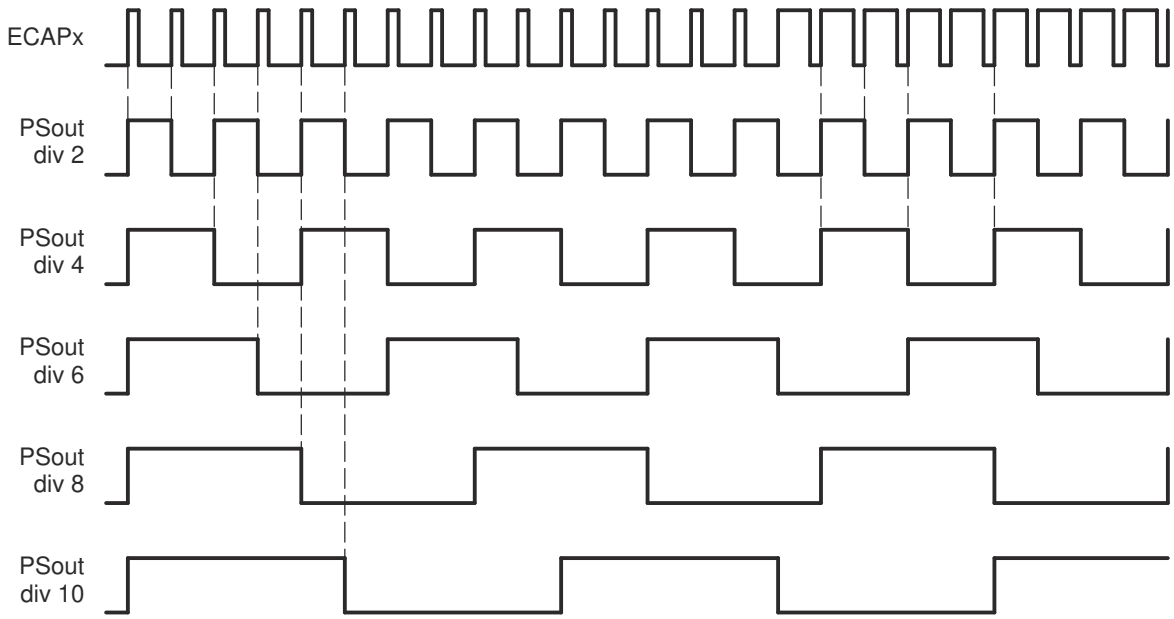



**Figure 16-5. Prescale Function Waveforms**

### 16.5.2 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.



Copyright © 2024 Texas Instruments Incorporated

- The edge event is gated to the respective CAPx register by the Mod4 counter. The CAPx register is loaded on the falling edge.

### *16.5.3 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.

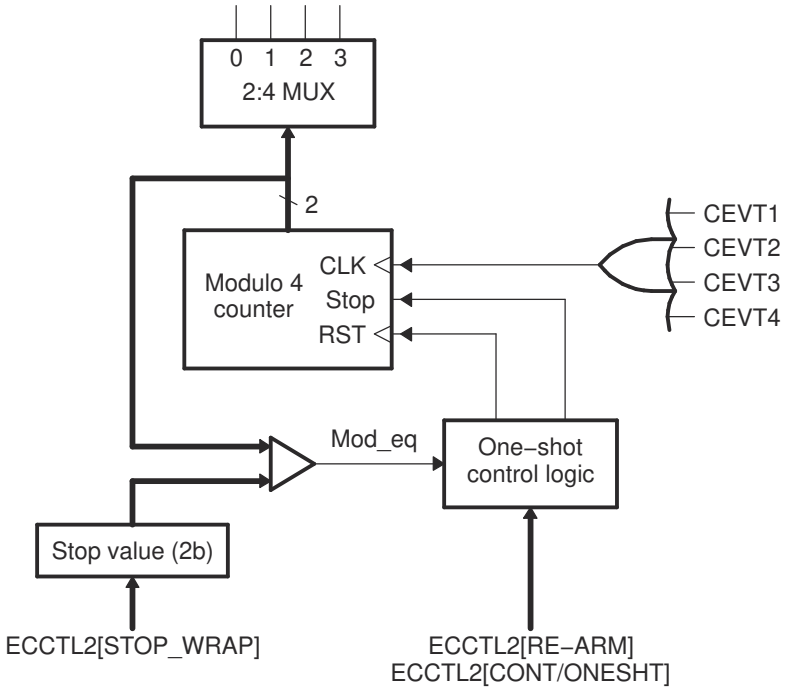



**Figure 16-6. Details of the Continuous/One-shot Block**



Copyright © 2024 Texas Instruments Incorporated

### 16.5.4 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.

### 16.5.5 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.

### 16.5.6 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 EPWM, eCAP, or X-Bar. The SWSYNC of the eCAP module is logical ORed with the SYNC signal as shown in Figure 16-7. The SYNC signal is defined by the selection of SYNCSELECT[ECAPxSYNCIN] as shown in Figure 16-8.

**Note**

If SWSYNC is forced from ECAPx, then the Sync pulse is delayed by one clock cycle for sync pulse receiving ECAPs . This cycle delay/lag for the sync pulse receiving ECAPs can be corrected with the CTRPHS register. If there is an external SYNC signal, then all ECAPs see the signal at the same time.



Copyright © 2024 Texas Instruments Incorporated

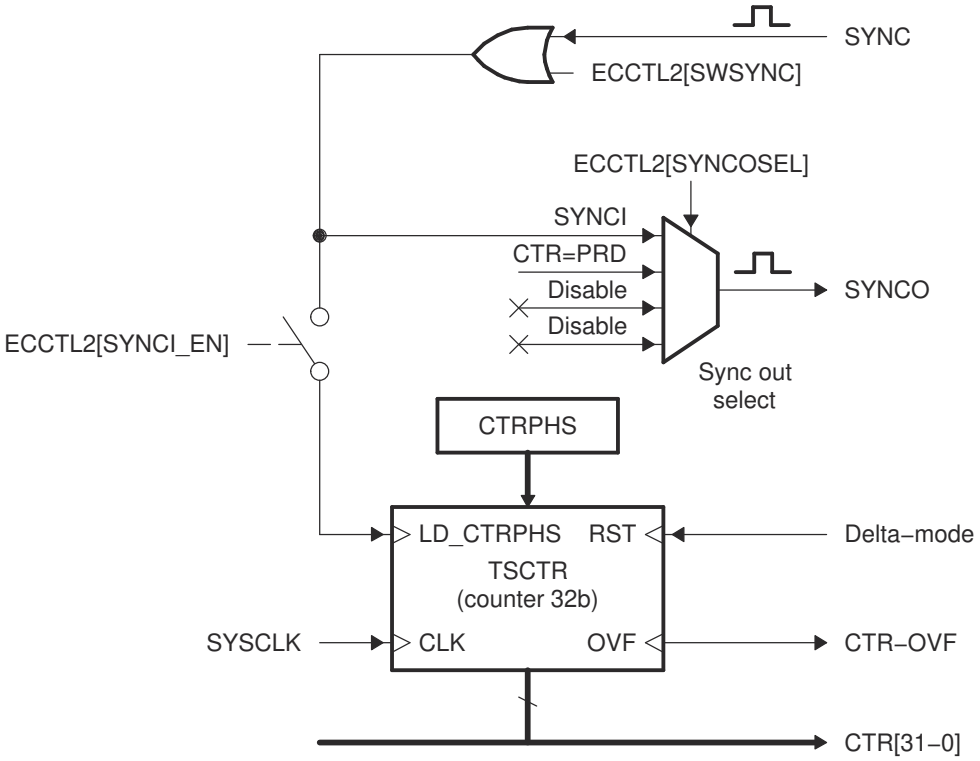



**Figure 16-7. Details of the Counter and Synchronization Block**

#### 16.5.6.1 Example 1 - Using SWSYNC with ECAP Module

Implement the following steps to use SWSYNC with ECAP1 and ECAP3.

- Configure ECAP[1..3].ECCTL2.SYNCO_SEL = 0x0, to allow the sync-in event to be the sync-out signal pass through.
- Configure ECAP[2..3].ECCTL2.SWSYNC = 0x0, to disable software synchronization for eCAP2 through eCAP3.
- The default sync signal comes from ePWM1, if TBCTL[SYNCOSEL] is not correctly configured this can cause undesired resets of the time-stamp register (TSCTR). Select an unused GPIO in InputXbarRegs.INPUT5SELECT. Configure this GPIO in output mode and write 0 to the GPIO DAT register. By default, this is programmed to GPIO0 so any activity on this pin can cause problems with the SWSYNC.
- Program SYNCSEL[ECAP1SYNCIN] = 0x5. This takes ECAPx.EXTSYNCIN to an inactive state.
- Configure ECAP1.ECCTL2.SWSYNC=0x1, this forces Software Synchronization of 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.



Copyright © 2024 Texas Instruments Incorporated

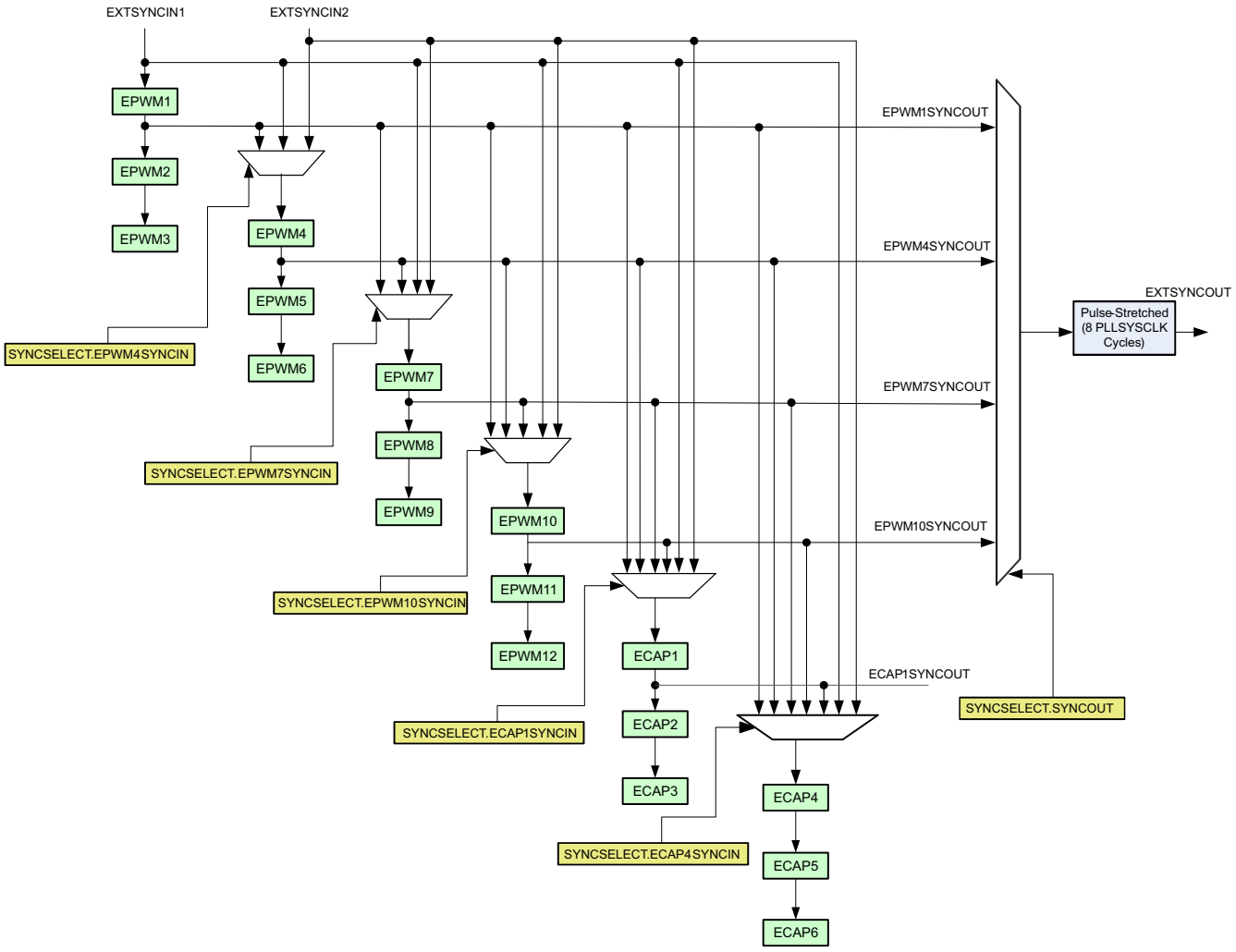



**Figure 16-8. Time-Base Counter Synchronization Scheme**

### *16.5.7 Interrupt Control*

Operation and features of the eCAP interrupt control include (see Figure 16-9):

- 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 and CLA.
- Seven interrupt events (CEVT1, CEVT2, CEVT3, CEVT4, CNTOVF, CTR=PRD, CTR=CMP) can be generated.
- The interrupt enable register (ECEINT) is used to enable/disable individual interrupt event sources. The interrupt flag register (ECFLG) indicates if any interrupt event has been latched and contains the global interrupt flag bit (INT). An interrupt pulse is generated to the PIE only if any of the interrupt events are enabled, the flag bit is 1, and the INT flag bit is 0. The interrupt service routine must clear the global interrupt flag bit and the serviced event using the interrupt clear register (ECCLR) before any other interrupt pulses are generated. To force an interrupt event, use the interrupt force register (ECFRC). This is useful for test purposes.



Copyright © 2024 Texas Instruments Incorporated

**Note**

The CEVT1, CEVT2, CEVT3, CEVT4 flags are only active in capture mode (ECCTL2[CAP/APWM == 0]). The CTR=PRD, CTR=CMP flags are only valid in APWM mode (ECCTL2[CAP/APWM == 1]). CNTOVF flag is valid in both modes.

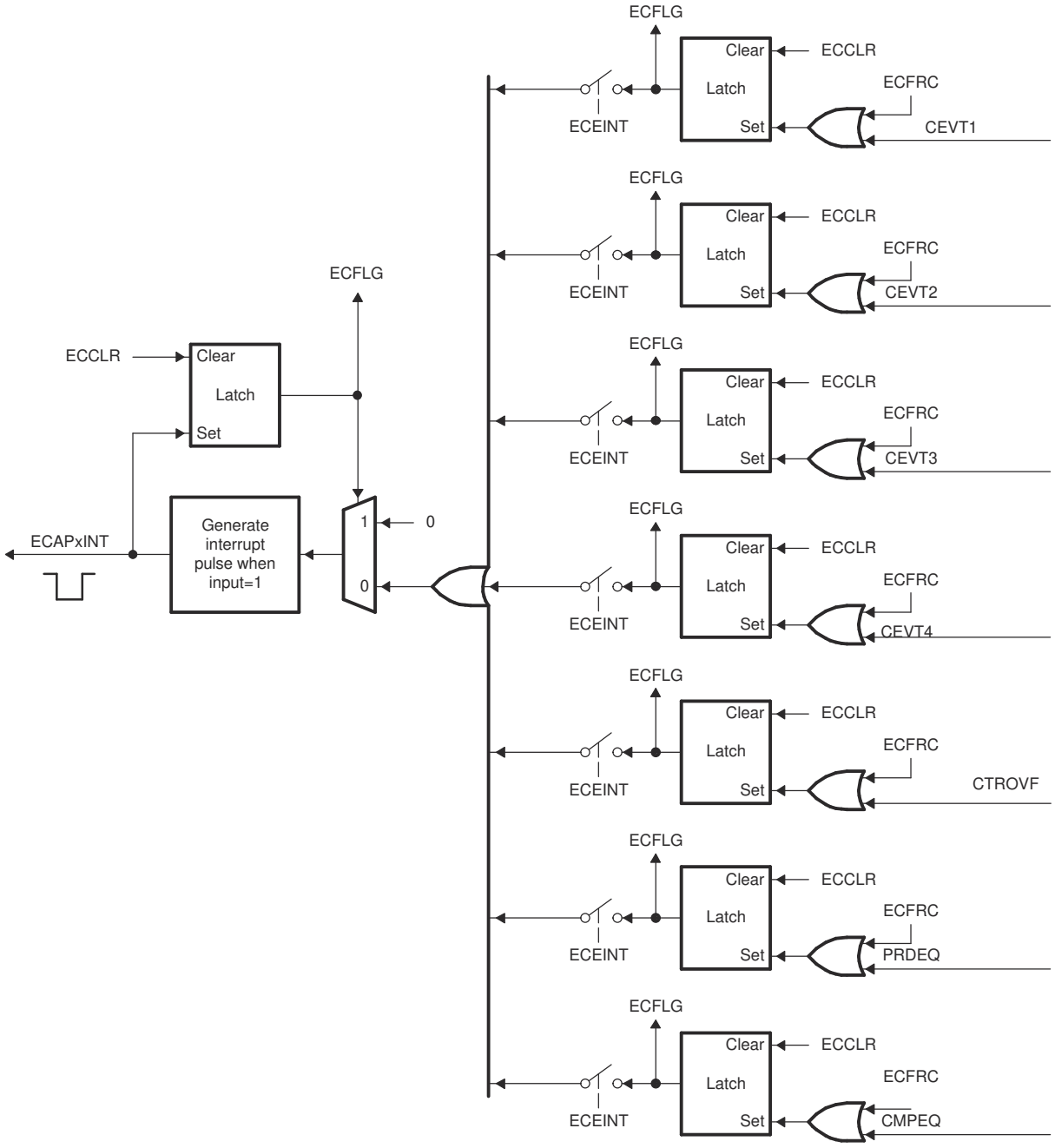



**Figure 16-9. Interrupts in eCAP Module**



Copyright © 2024 Texas Instruments Incorporated

### 16.5.8 DMA Interrupt

On Type 0 eCAP modules, the CPU was required to begin data transfers using DMA. New to the Type 1 eCAP, a separate DMA Trigger (ECAP_DMA_INT) enables continuous transfer of capture data from eCAP registers to on-chip memory using DMA. Any one of the four available interrupt events () can be selected as the trigger source for ECAP_DMA_INT using ECCTL2 [DMAEVTSEL].

### 16.5.9 Shadow Load and Lockout Control

In capture mode, this logic inhibits (locks out) any shadow loading of CAP1 or CAP2 from APRD and ACMP registers, respectively.

In APWM mode, shadow loading is active and two choices are permitted:

- Immediate - APRD or ACMP are transferred to CAP1 or CAP2 immediately upon writing a new value.
- On period equal, CTR[31:0] = PRD[31:0].

### 16.5.10 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.

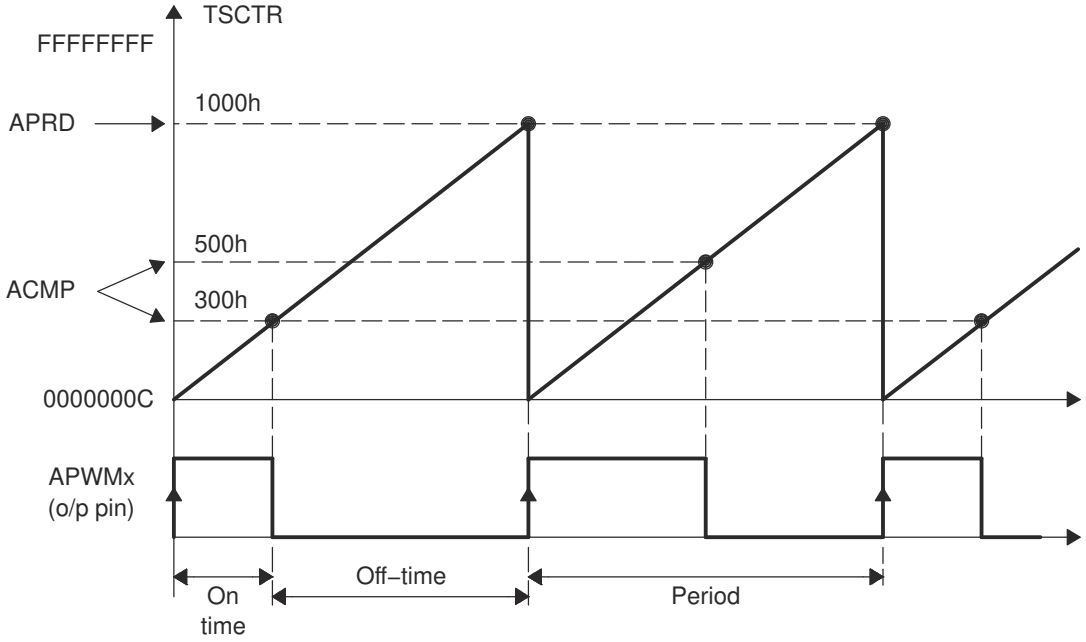



**Figure 16-10. PWM Waveform Details Of APWM Mode Operation**



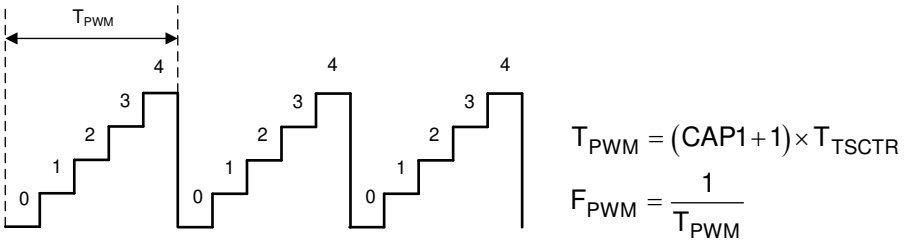
Copyright © 2024 Texas Instruments Incorporated

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
```




$$T_{PWM} = (CAP1 + 1) \times T_{TSCTR}$$

$$F_{PWM} = \frac{1}{T_{PWM}}$$

**Figure 16-11. Time-Base Frequency and Period Calculation**



Copyright © 2024 Texas Instruments Incorporated

## 16.6 Application of the eCAP Module

The following sections provide applications examples to show how to operate the eCAP module.

### *16.6.1 Example 1 - Absolute Time-Stamp Operation Rising-Edge Trigger*

Figure 16-12 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 16-12), 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.

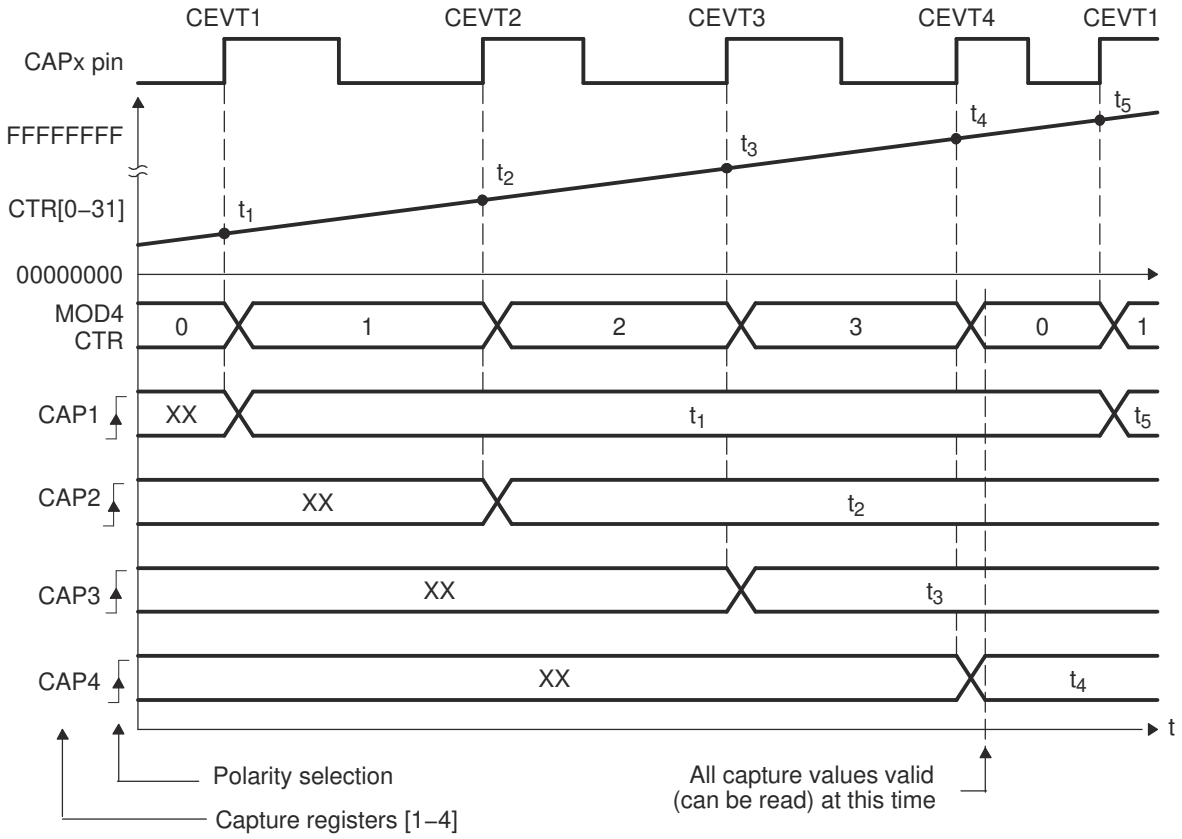



**Figure 16-12. Capture Sequence for Absolute Time-stamp and Rising-Edge Detect**



Copyright © 2024 Texas Instruments Incorporated

### 16.6.2 Example 2 - Absolute Time-Stamp Operation Rising- and Falling-Edge Trigger

In Figure 16-13, 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: Period1 = $t_3 - t_1$, Period2 = $t_5 - t_3$, …and so on. Duty Cycle1 (on-time %) = $(t_2 - t_1)$ / Period1 x 100%, and so on. Duty Cycle1 (off-time %) = $(t_3 - t_2)$ / Period1 x 100%, and so on.

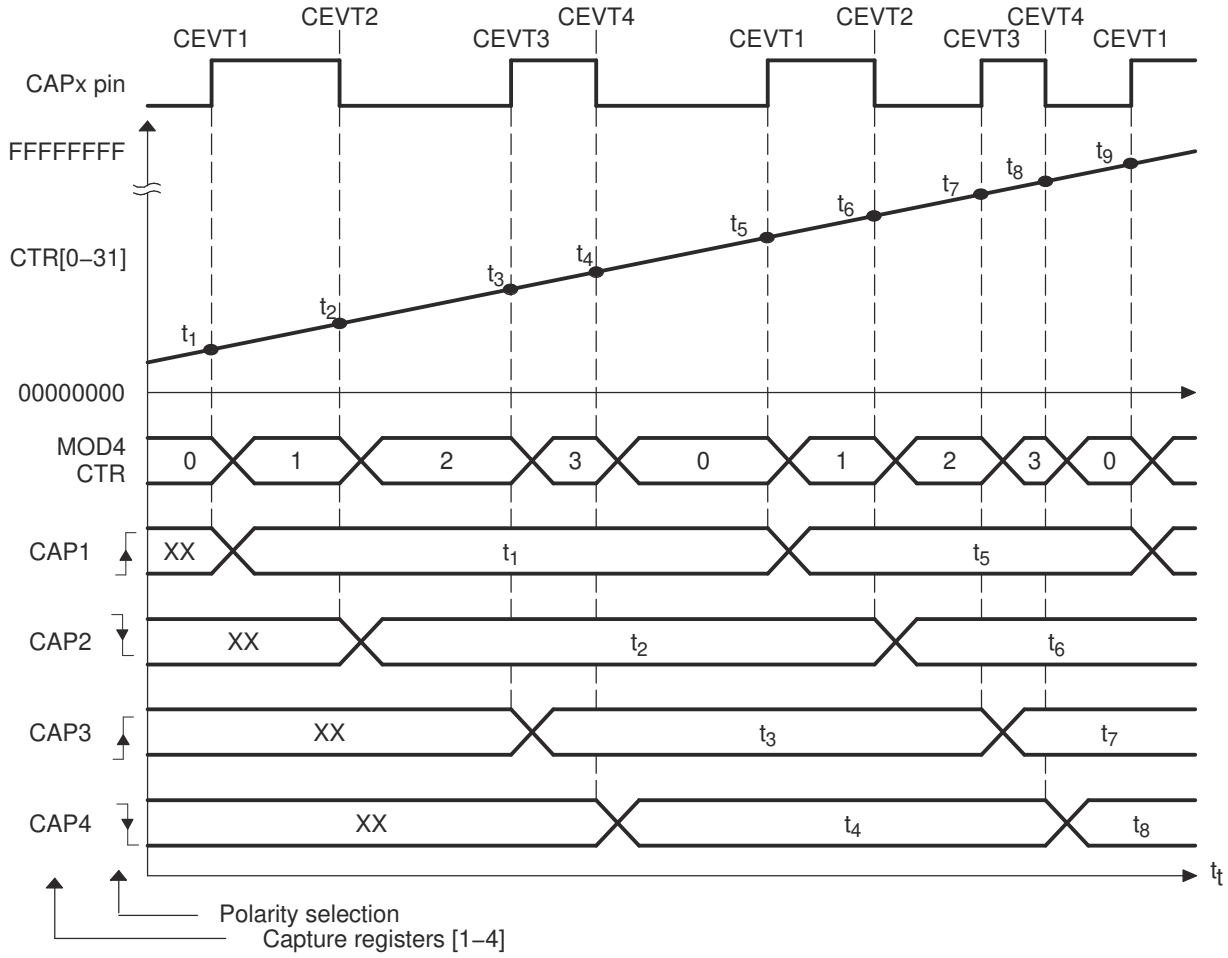



**Figure 16-13. Capture Sequence for Absolute Time-stamp with Rising- and Falling-Edge Detect**



Copyright © 2024 Texas Instruments Incorporated

### 16.6.3 Example 3 - Time Difference (Delta) Operation Rising-Edge Trigger

Figure 16-14 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 16-14, 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.

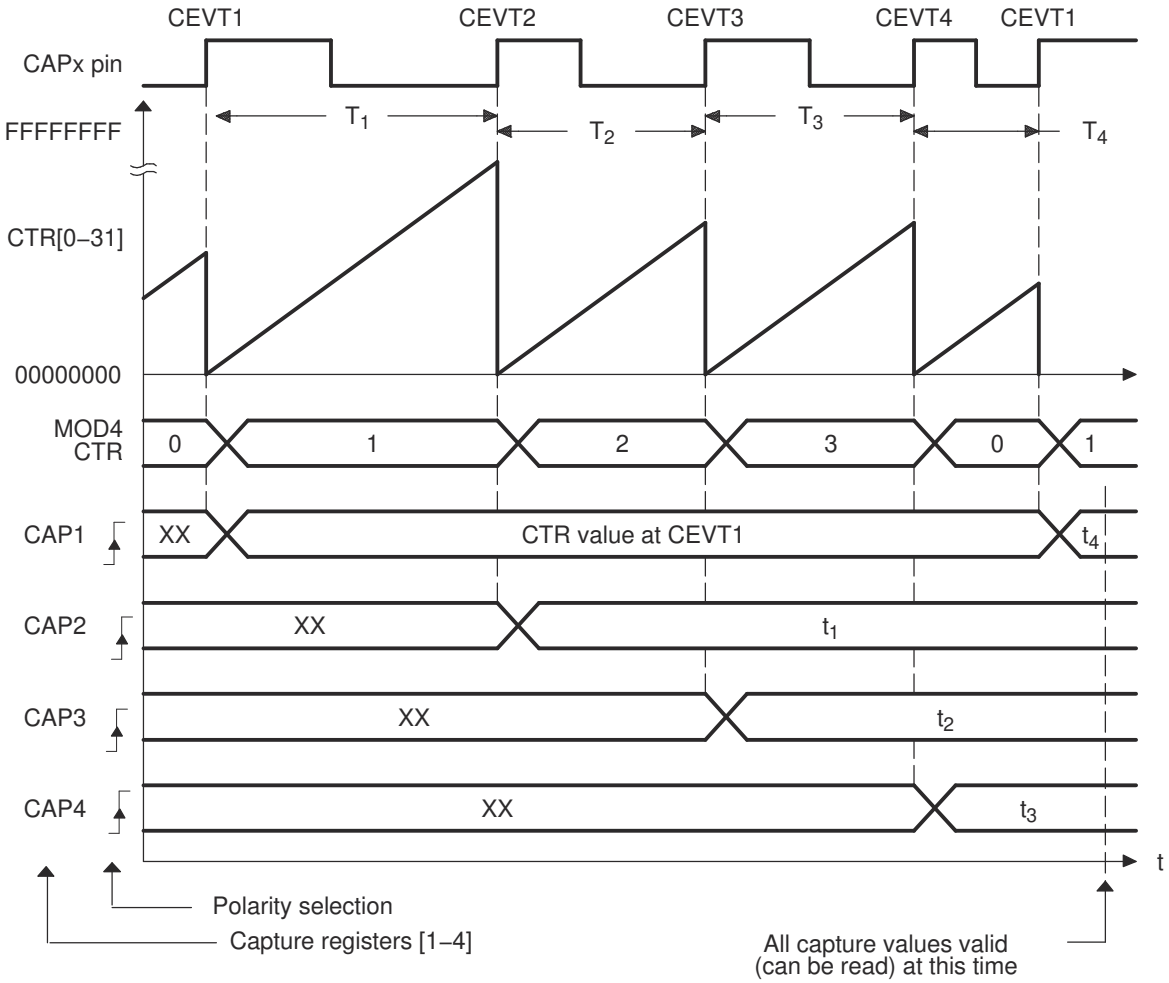



**Figure 16-14. Capture Sequence for Delta Mode Time-stamp and Rising Edge Detect**



Copyright © 2024 Texas Instruments Incorporated

### 16.6.4 Example 4 - Time Difference (Delta) Operation Rising- and Falling-Edge Trigger

In Figure 16-15, the eCAP operating mode is almost the same as in previous section except capture events are qualified as either rising or falling edge, this now gives both period and duty cycle information, that is: Period1 = $T_1$+$T_2$, Period2 = $T_3$+$T_4$, and so on. Duty Cycle1 (on-time %) = $T_1$ / Period1 x 100%, Duty Cycle1 (off-time %) = $T_2$ / Period1 x 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.

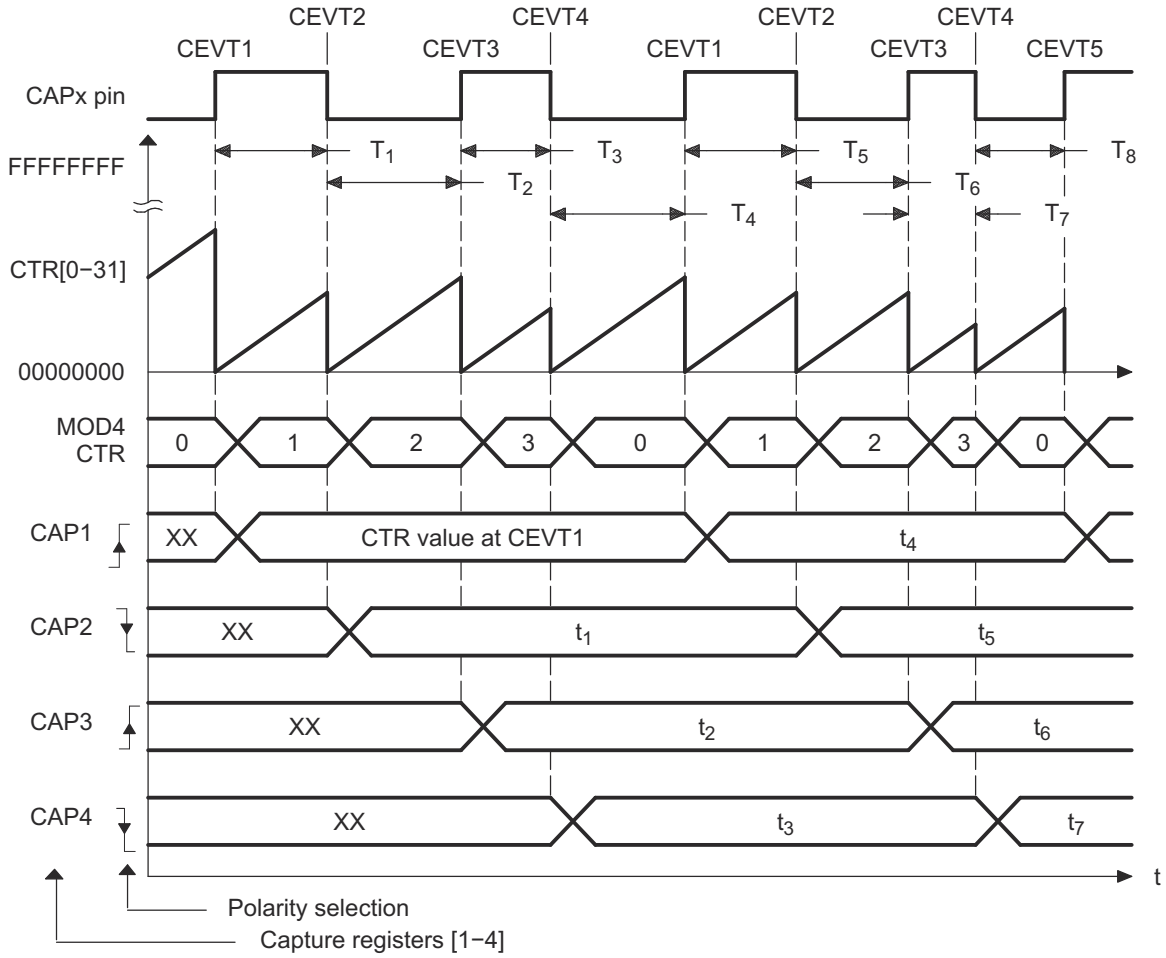



**Figure 16-15. Capture Sequence for Delta Mode Time-stamp with Rising- and Falling-Edge Detect**



Copyright © 2024 Texas Instruments Incorporated

## 16.7 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.

### *16.7.1 Example 1 - Simple PWM Generation (Independent Channels)*

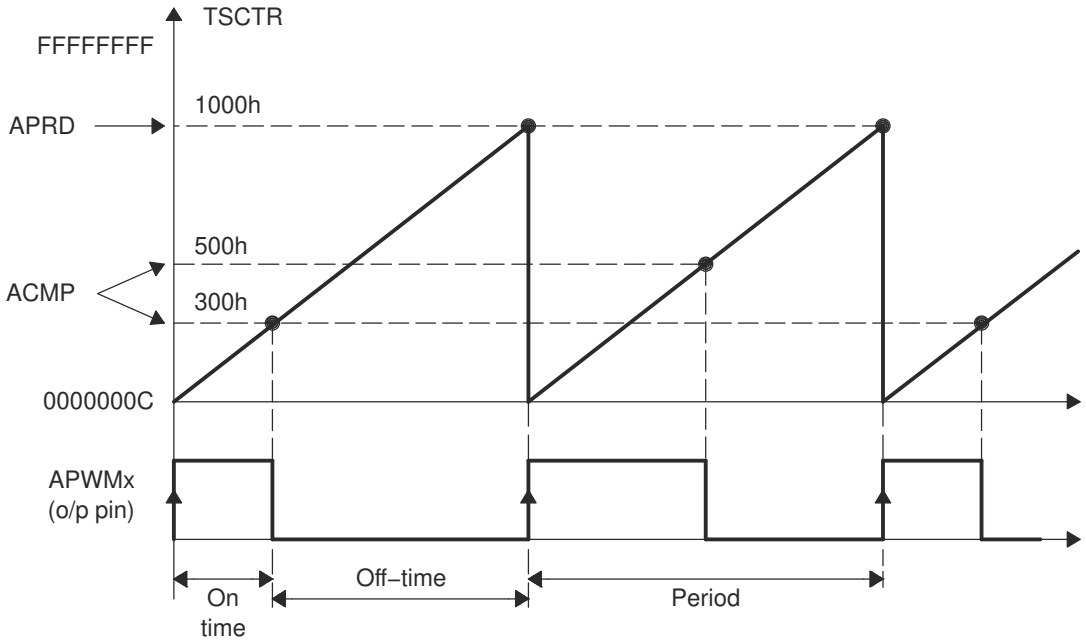



**Figure 16-16. PWM Waveform Details of APWM Mode Operation**



Copyright © 2024 Texas Instruments Incorporated

## 16.8 Software

### *16.8.1 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.

#### 16.8.1.1 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.

#### 16.8.1.2 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.

#### 16.8.1.3 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.

#### 16.8.1.4 eCAP Software Sync Example

FILE: ecap_ex4_sw_sync.c

This example configures ePWM3A for:
- Up count mode
- Period starts at 500 and goes up to 8000
- Toggle output on PRD

eCAP1, eCAP2 and eCAP3 are configured to capture the time between rising and falling edge of the ePWM3A output.

*External Connections*
- eCAP1, eCAP2, eCAP3 are on GPIO16
- ePWM3A is on GPIO4
- Connect GPIO4 to GPIO16.



Copyright © 2024 Texas Instruments Incorporated

*Watch Variables*

- *ecapPassCount* - Successful captures.
- *ecap3IntCount* - Interrupt counts.

## 16.9 eCAP Registers

This section describes the Enhanced Capture Registers.

### *16.9.1 eCAP Base Addresses*

**Table 16-1. eCAP Base Address Table**

| Device Register | Register Name | Start Address | End Address |
|---|---|---|---|
| ECap1Regs | ECAP_REGS | 0x0000_5000 | 0x0000_501F |
| ECap2Regs | ECAP_REGS | 0x0000_5020 | 0x0000_503F |
| ECap3Regs | ECAP_REGS | 0x0000_5040 | 0x0000_505F |
| ECap4Regs | ECAP_REGS | 0x0000_5060 | 0x0000_507F |
| ECap5Regs | ECAP_REGS | 0x0000_5080 | 0x0000_509F |
| ECap6Regs | ECAP_REGS | 0x0000_50A0 | 0x0000_50BF |



Copyright © 2024 Texas Instruments Incorporated

### 16.9.2 ECAP_REGS Registers

Table 16-2 lists the memory-mapped registers for the ECAP_REGS registers. All register offset addresses not listed in Table 16-2 should be considered as reserved locations and the register contents should not be modified.

**Table 16-2. ECAP_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | TSCTR | Time-Stamp Counter | | Go |
| 2h | CTRPHS | Counter Phase Offset Value Register | | Go |
| 4h | CAP1 | Capture 1 Register | | Go |
| 6h | CAP2 | Capture 2 Register | | Go |
| 8h | CAP3 | Capture 3 Register | | Go |
| Ah | CAP4 | Capture 4 Register | | Go |
| 14h | ECCTL1 | Capture Control Register 1 | | Go |
| 15h | ECCTL2 | Capture Control Register 2 | | Go |
| 16h | ECEINT | Capture Interrupt Enable Register | | Go |
| 17h | ECFLG | Capture Interrupt Flag Register | | Go |
| 18h | ECCLR | Capture Interrupt Clear Register | | Go |
| 19h | ECFRC | Capture Interrupt Force Register | | Go |

Complex bit access types are encoded to fit into small table cells. Table 16-3 shows the codes that are used for access types in this section.

**Table 16-3. ECAP_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| R-0 | R<br>-0 | Read<br>Returns 0s |
| Write Type | | |
| W | W | Write |
| W1C | W<br>1C | Write<br>1 to clear |
| W1S | W<br>1S | Write<br>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. |



Copyright © 2024 Texas Instruments Incorporated

#### 16.9.2.1 TSCTR Register (Offset = 0h) [Reset = 00000000h]

TSCTR is shown in Figure 16-17 and described in Table 16-4.

Return to the Summary Table.

Time-Stamp Counter

**Figure 16-17. 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 16-4. 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<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 16.9.2.2 CTRPHS Register (Offset = 2h) [Reset = 00000000h]

CTRPHS is shown in Figure 16-18 and described in Table 16-5.

Return to the Summary Table.

Counter Phase Offset Value Register

**Figure 16-18. 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 16-5. 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 timebases.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 16.9.2.3 CAP1 Register (Offset = 4h) [Reset = 00000000h]

CAP1 is shown in Figure 16-19 and described in Table 16-6.

Return to the Summary Table.

Capture 1 Register

**Figure 16-19. 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 16-6. CAP1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | CAP1 | R/W | 0h | This register can be loaded (written) by:<br>- Time-Stamp counter value (TSCTR) during a capture event<br>- Software - may be useful for test purposes or initialization<br>- ARPD shadow register (CAP3) when used in APWM mode<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 16.9.2.4 CAP2 Register (Offset = 6h) [Reset = 00000000h]

CAP2 is shown in Figure 16-20 and described in Table 16-7.

Return to the Summary Table.

Capture 2 Register

**Figure 16-20. 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 16-7. CAP2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | CAP2 | R/W | 0h | This register can be loaded (written) by:<br>- Time-Stamp ( counter value) during a capture event<br>- Software - may be useful for test purposes<br>- ACMP shadow register (CAP4) when used in APWM mode<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 16.9.2.5 CAP3 Register (Offset = 8h) [Reset = 00000000h]

CAP3 is shown in Figure 16-21 and described in Table 16-8.

Return to the Summary Table.

Capture 3 Register

**Figure 16-21. 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 16-8. 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.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 16.9.2.6 CAP4 Register (Offset = Ah) [Reset = 00000000h]

CAP4 is shown in Figure 16-22 and described in Table 16-9.

Return to the Summary Table.

Capture 4 Register

**Figure 16-22. 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 16-9. 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.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 16.9.2.7 ECCTL1 Register (Offset = 14h) [Reset = 0000h]

ECCTL1 is shown in Figure 16-23 and described in Table 16-10.

Return to the Summary Table.

Capture Control Register 1

**Figure 16-23. 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 16-10. ECCTL1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-14 | FREE_SOFT | R/W | 0h | Emulation Control<br>Reset type: SYSRSn<br>0h (R/W) = TSCTR counter stops immediately on emulation suspend<br>1h (R/W) = TSCTR counter runs until = 0<br>2h (R/W) = TSCTR counter is unaffected by emulation suspend (Run Free)<br>3h (R/W) = TSCTR counter is unaffected by emulation suspend (Run Free) |
| 13-9 | PRESCALE | R/W | 0h | Event Filter prescale select<br>Reset type: SYSRSn<br>0h (R/W) = Divide by 1 (i.e,. no prescale, by-pass the prescaler)<br>1h (R/W) = Divide by 2<br>2h (R/W) = Divide by 4<br>3h (R/W) = Divide by 6<br>4h (R/W) = Divide by 8<br>5h (R/W) = Divide by 10<br>1Eh (R/W) = Divide by 60<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = Disable CAP1-4 register loads at capture event time.<br>1h (R/W) = Enable CAP1-4 register loads at capture event time. |
| 7 | CTRRST4 | R/W | 0h | Counter Reset on Capture Event 4<br>Reset type: SYSRSn<br>0h (R/W) = Do not reset counter on Capture Event 4 (absolute time stamp operation)<br>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<br>Reset type: SYSRSn<br>0h (R/W) = Capture Event 4 triggered on a rising edge (RE)<br>1h (R/W) = Capture Event 4 triggered on a falling edge (FE) |
| 5 | CTRRST3 | R/W | 0h | Counter Reset on Capture Event 3<br>Reset type: SYSRSn<br>0h (R/W) = Do not reset counter on Capture Event 3 (absolute time stamp)<br>1h (R/W) = Reset counter after Event 3 time-stamp has been captured (used in difference mode operation) |



Copyright © 2024 Texas Instruments Incorporated

**Table 16-10. ECCTL1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 4 | CAP3POL | R/W | 0h | Capture Event 3 Polarity select<br>Reset type: SYSRSn<br>0h (R/W) = Capture Event 3 triggered on a rising edge (RE)<br>1h (R/W) = Capture Event 3 triggered on a falling edge (FE) |
| 3 | CTRRST2 | R/W | 0h | Counter Reset on Capture Event 2<br>Reset type: SYSRSn<br>0h (R/W) = Do not reset counter on Capture Event 2 (absolute time stamp)<br>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<br>Reset type: SYSRSn<br>0h (R/W) = Capture Event 2 triggered on a rising edge (RE)<br>1h (R/W) = Capture Event 2 triggered on a falling edge (FE) |
| 1 | CTRRST1 | R/W | 0h | Counter Reset on Capture Event 1<br>Reset type: SYSRSn<br>0h (R/W) = Do not reset counter on Capture Event 1 (absolute time stamp)<br>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<br>Reset type: SYSRSn<br>0h (R/W) = Capture Event 1 triggered on a rising edge (RE)<br>1h (R/W) = Capture Event 1 triggered on a falling edge (FE) |



Copyright © 2024 Texas Instruments Incorporated

### 16.9.2.8 ECCTL2 Register (Offset = 15h) [Reset = 0006h]

ECCTL2 is shown in Figure 16-24 and described in Table 16-11.

Return to the Summary Table.

Capture Control Register 2

**Figure 16-24. ECCTL2 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | APWMPOL | CAP_APWM | SWSYNC |
| R-0h | | | | | R/W-0h | R/W-0h | R-0/W1S-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SYNCO_SEL | | SYNCI_EN | TSCTRSTOP | REARM | STOP_WRAP | | CONT_ONESHT |
| R/W-0h | | R/W-0h | R/W-0h | R-0/W1S-0h | R/W-3h | | R/W-0h |

**Table 16-11. ECCTL2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-11 | RESERVED | R | 0h | Reserved |
| 10 | APWMPOL | R/W | 0h | APWM output polarity select. This is applicable only in APWM operating mode.<br>Reset type: SYSRSn<br>0h (R/W) = Output is active high (Compare value defines high time)<br>1h (R/W) = Output is active low (Compare value defines low time) |
| 9 | CAP_APWM | R/W | 0h | CAP/APWM operating mode select<br>Reset type: SYSRSn<br>0h (R/W) = ECAP module operates in capture mode. This mode forces the following configuration:<br>- Inhibits TSCTR resets via CTR = PRD event<br>- Inhibits shadow loads on CAP1 and 2 registers<br>- Permits user to enable CAP1-4 register load<br>- CAPx/APWMx pin operates as a capture input<br>1h (R/W) = ECAP module operates in APWM mode. This mode forces the following configuration:<br>- Resets TSCTR on CTR = PRD event (period boundary<br>- Permits shadow loading on CAP1 and 2 registers<br>- Disables loading of time-stamps into CAP1-4 registers<br>- CAPx/APWMx pin operates as a APWM output |
| 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.<br>Reset type: SYSRSn<br>0h (R/W) = Writing a zero has no effect. Reading always returns a zero<br>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<br>Reset type: SYSRSn<br>0h (R/W) = Select sync-in event to be the sync-out signal (pass through)<br>1h (R/W) = Select CTR = PRD event to be the sync-out signal. Note: Selection CTR = PRD is meaningful only in APWM mode<br>2h (R/W) = Disable sync out signal<br>3h (R/W) = Disable sync out signal |



Copyright © 2024 Texas Instruments Incorporated

**Table 16-11. ECCTL2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 5 | SYNCI_EN | R/W | 0h | Counter (TSCTR) Sync-In select mode<br>Reset type: SYSRSn<br>0h (R/W) = Disable sync-in option<br>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<br>Reset type: SYSRSn<br>0h (R/W) = TSCTR stopped<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = Has no effect (reading always returns a 0)<br>1h (R/W) = Arms the one-shot sequence as follows:<br>1) Resets the Mod4 counter to zero<br>2) Unfreezes the Mod4 counter<br>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.<br>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.<br>Notes: STOP_WRAP is compared to Mod4 counter and, when equal, 2 actions occur:<br>- Mod4 counter is stopped (frozen)<br>- Capture register loads are inhibited<br>In one-shot mode, further interrupt events are blocked until re-armed.<br>Reset type: SYSRSn<br>0h (R/W) = Stop after Capture Event 1 in one-shot mode<br>Wrap after Capture Event 1 in continuous mode.<br>1h (R/W) = Stop after Capture Event 2 in one-shot mode<br>Wrap after Capture Event 2 in continuous mode.<br>2h (R/W) = Stop after Capture Event 3 in one-shot mode<br>Wrap after Capture Event 3 in continuous mode.<br>3h (R/W) = Stop after Capture Event 4 in one-shot mode<br>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)<br>Reset type: SYSRSn<br>0h (R/W) = Operate in continuous mode<br>1h (R/W) = Operate in one-Shot mode |



Copyright © 2024 Texas Instruments Incorporated

### 16.9.2.9 ECEINT Register (Offset = 16h) [Reset = 0000h]

ECEINT is shown in Figure 16-25 and described in Table 16-12.

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 16-25. ECEINT Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CTR_EQ_CMP | CTR_EQ_PRD | CTROVF | CEVT4 | CEVT3 | CEVT2 | CEVT1 | RESERVED |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R-0h |

**Table 16-12. ECEINT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R | 0h | Reserved |
| 7 | CTR_EQ_CMP | R/W | 0h | Counter Equal Compare Interrupt Enable<br>Reset type: SYSRSn<br>0h (R/W) = Disable Compare Equal as an Interrupt source<br>1h (R/W) = Enable Compare Equal as an Interrupt source |
| 6 | CTR_EQ_PRD | R/W | 0h | Counter Equal Period Interrupt Enable<br>Reset type: SYSRSn<br>0h (R/W) = Disable Period Equal as an Interrupt source<br>1h (R/W) = Enable Period Equal as an Interrupt source |
| 5 | CTROVF | R/W | 0h | Counter Overflow Interrupt Enable<br>Reset type: SYSRSn<br>0h (R/W) = Disabled counter Overflow as an Interrupt source<br>1h (R/W) = Enable counter Overflow as an Interrupt source |
| 4 | CEVT4 | R/W | 0h | Capture Event 4 Interrupt Enable<br>Reset type: SYSRSn<br>0h (R/W) = Disable Capture Event 4 as an Interrupt source<br>1h (R/W) = Capture Event 4 Interrupt Enable |
| 3 | CEVT3 | R/W | 0h | Capture Event 3 Interrupt Enable<br>Reset type: SYSRSn<br>0h (R/W) = Disable Capture Event 3 as an Interrupt source<br>1h (R/W) = Enable Capture Event 3 as an Interrupt source |
| 2 | CEVT2 | R/W | 0h | Capture Event 2 Interrupt Enable<br>Reset type: SYSRSn<br>0h (R/W) = Disable Capture Event 2 as an Interrupt source<br>1h (R/W) = Enable Capture Event 2 as an Interrupt source |



Copyright © 2024 Texas Instruments Incorporated

**Table 16-12. ECEINT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1 | CEVT1 | R/W | 0h | Capture Event 1 Interrupt Enable<br>Reset type: SYSRSn<br>0h (R/W) = Disable Capture Event 1 as an Interrupt source<br>1h (R/W) = Enable Capture Event 1 as an Interrupt source |
| 0 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 16.9.2.10 ECFLG Register (Offset = 17h) [Reset = 0000h]

ECFLG is shown in Figure 16-26 and described in Table 16-13.

Return to the Summary Table.

Capture Interrupt Flag Register

**Figure 16-26. ECFLG Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CTR_CMP | CTR_PRD | CTROVF | CEVT4 | CEVT3 | CEVT2 | CEVT1 | INT |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 16-13. ECFLG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R | 0h | Reserved |
| 7 | CTR_CMP | R | 0h | Compare Equal Compare Status Flag. This flag is active only in APWM mode.<br>Reset type: SYSRSn<br>0h (R/W) = Indicates no event occurred<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = Indicates no event occurred<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = Indicates no event occurred<br>1h (R/W) = Indicates the counter (TSCTR) has made the transition from FFFFFFFF ' 00000000 |
| 4 | CEVT4 | R | 0h | Capture Event 4 Status Flag This flag is only active in CAP mode.<br>Reset type: SYSRSn<br>0h (R/W) = Indicates no event occurred<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = Indicates no event occurred<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = Indicates no event occurred<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = Indicates no event occurred<br>1h (R/W) = Indicates the first event occurred at ECAPx pin. |
| 0 | INT | R | 0h | Global Interrupt Status Flag<br>Reset type: SYSRSn<br>0h (R/W) = Indicates no event occurred<br>1h (R/W) = Indicates that an interrupt was generated. |



Copyright © 2024 Texas Instruments Incorporated

### 16.9.2.11 ECCLR Register (Offset = 18h) [Reset = 0000h]

ECCLR is shown in Figure 16-27 and described in Table 16-14.

Return to the Summary Table.

Capture Interrupt Clear Register

**Figure 16-27. ECCLR Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CTR_CMP | CTR_PRD | CTROVF | CEVT4 | CEVT3 | CEVT2 | CEVT1 | INT |
| R-0/W1C-0h | R-0/W1C-0h | R-0/W1C-0h | R-0/W1C-0h | R-0/W1C-0h | R-0/W1C-0h | R-0/W1C-0h | R-0/W1C-0h |

**Table 16-14. ECCLR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R | 0h | Reserved |
| 7 | CTR_CMP | R-0/W1C | 0h | Counter Equal Compare Status Clear<br>Reset type: SYSRSn<br>0h (R/W) = Writing a 0 has no effect. Always reads back a 0<br>1h (R/W) = Writing a 1 clears the CTR=CMP flag. |
| 6 | CTR_PRD | R-0/W1C | 0h | Counter Equal Period Status Clear<br>Reset type: SYSRSn<br>0h (R/W) = Writing a 0 has no effect. Always reads back a 0<br>1h (R/W) = Writing a 1 clears the CTR=PRD flag. |
| 5 | CTROVF | R-0/W1C | 0h | Counter Overflow Status Clear<br>Reset type: SYSRSn<br>0h (R/W) = Writing a 0 has no effect. Always reads back a 0<br>1h (R/W) = Writing a 1 clears the CTROVF flag. |
| 4 | CEVT4 | R-0/W1C | 0h | Capture Event 4 Status Clear<br>Reset type: SYSRSn<br>0h (R/W) = Writing a 0 has no effect. Always reads back a 0<br>1h (R/W) = Writing a 1 clears the CEVT4 flag. |
| 3 | CEVT3 | R-0/W1C | 0h | Capture Event 3 Status Clear<br>Reset type: SYSRSn<br>0h (R/W) = Writing a 0 has no effect. Always reads back a 0<br>1h (R/W) = Writing a 1 clears the CEVT3 flag. |
| 2 | CEVT2 | R-0/W1C | 0h | Capture Event 2 Status Clear<br>Reset type: SYSRSn<br>0h (R/W) = Writing a 0 has no effect. Always reads back a 0<br>1h (R/W) = Writing a 1 clears the CEVT2 flag. |
| 1 | CEVT1 | R-0/W1C | 0h | Capture Event 1 Status Clear<br>Reset type: SYSRSn<br>0h (R/W) = Writing a 0 has no effect. Always reads back a 0<br>1h (R/W) = Writing a 1 clears the CEVT1 flag. |
| 0 | INT | R-0/W1C | 0h | ECAP Global Interrupt Status Clear<br>Reset type: SYSRSn<br>0h (R/W) = Writing a 0 has no effect. Always reads back a 0<br>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 |



Copyright © 2024 Texas Instruments Incorporated

### 16.9.2.12 ECFRC Register (Offset = 19h) [Reset = 0000h]

ECFRC is shown in Figure 16-28 and described in Table 16-15.

Return to the Summary Table.

Capture Interrupt Force Register

**Figure 16-28. ECFRC Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CTR_CMP | CTR_PRD | CTROVF | CEVT4 | CEVT3 | CEVT2 | CEVT1 | RESERVED |
| R-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 16-15. ECFRC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R | 0h | Reserved |
| 7 | CTR_CMP | R-0/W1S | 0h | Force Counter Equal Compare Interrupt. This event is only active in APWM mode.<br>Reset type: SYSRSn<br>0h (R/W) = No effect. Always reads back a 0.<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = No effect. Always reads back a 0.<br>1h (R/W) = Writing a 1 sets the CTR=PRD flag. |
| 5 | CTROVF | R-0/W1S | 0h | Force Counter Overflow.<br>Reset type: SYSRSn<br>0h (R/W) = No effect. Always reads back a 0.<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = No effect. Always reads back a 0.<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = No effect. Always reads back a 0.<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = No effect. Always reads back a 0.<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = No effect. Always reads back a 0.<br>1h (R/W) = Sets the CEVT1 flag. |
| 0 | RESERVED | R | 0h | Reserved |

## *16.9.3 ECAP Registers to Driverlib Functions*

**Table 16-16. ECAP Registers to Driverlib Functions**

| File | Driverlib Function |
|---|---|
| **TSCTR** | |
| ecap.h | ECAP_getTimeBaseCounter |



Copyright © 2024 Texas Instruments Incorporated

**Table 16-16. ECAP Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| **CTRPHS** | |
| ecap.h | ECAP_setPhaseShiftCount |
| **CAP1** | |
| ecap.h | ECAP_setAPWMPeriod |
| ecap.h | ECAP_getEventTimeStamp |
| **CAP2** | |
| ecap.h | ECAP_setAPWMCompare |
| ecap.h | ECAP_getEventTimeStamp |
| **CAP3** | |
| ecap.h | ECAP_setAPWMShadowPeriod |
| ecap.h | ECAP_getEventTimeStamp |
| **CAP4** | |
| ecap.h | ECAP_setAPWMShadowCompare |
| ecap.h | ECAP_getEventTimeStamp |
| **ECCTL1** | |
| 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 |
| **ECCTL2** | |
| 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 |
| **ECEINT** | |
| ecap.h | ECAP_enableInterrupt |
| ecap.h | ECAP_disableInterrupt |
| **ECFLG** | |
| ecap.h | ECAP_getInterruptSource |
| ecap.h | ECAP_getGlobalInterruptStatus |
| **ECCLR** | |
| ecap.h | ECAP_clearInterrupt |
| ecap.h | ECAP_clearGlobalInterrupt |
| **ECFRC** | |



Copyright © 2024 Texas Instruments Incorporated

**Table 16-16. ECAP Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| ecap.h | ECAP_forceInterrupt |



Copyright © 2024 Texas Instruments Incorporated

Chapter 17

# Enhanced Quadrature Encoder Pulse (eQEP)

The enhanced Quadrature Encoder Pulse (eQEP) module described here is a Type 0 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.


**17.1 Introduction** ..... 2193
**17.2 Configuring Device Pins** ..... 2195
**17.3 Description** ..... 2196
**17.4 Quadrature Decoder Unit (QDU)** ..... 2199
**17.5 Position Counter and Control Unit (PCCU)** ..... 2202
**17.6 eQEP Edge Capture Unit** ..... 2210
**17.7 eQEP Watchdog** ..... 2214
**17.8 eQEP Unit Timer Base** ..... 2214
**17.9 eQEP Interrupt Structure** ..... 2215
**17.10 eQEP Registers** ..... 2215





Copyright © 2024 Texas Instruments Incorporated

## 17.1 Introduction

An incremental encoder disk is patterned with a track of slots along the periphery, as shown in Figure 17-1. 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

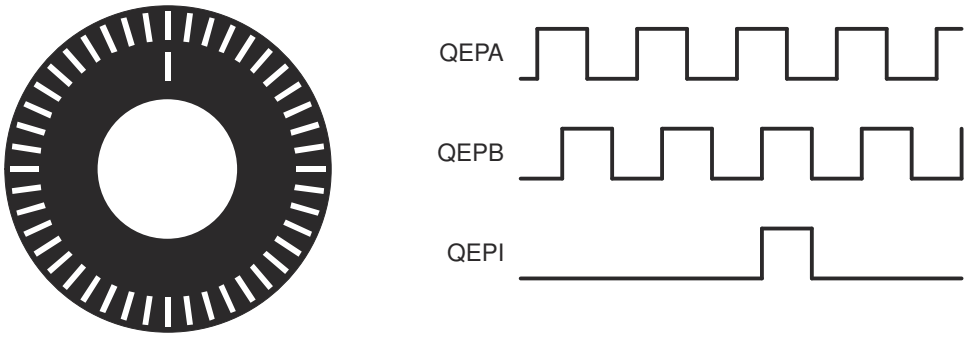
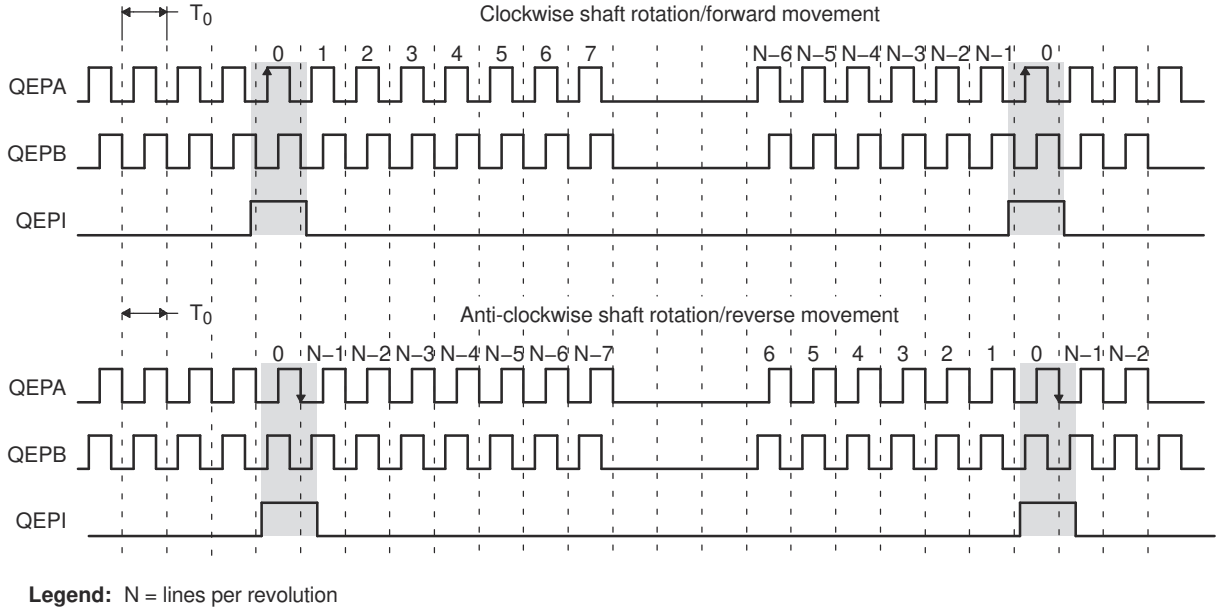



**Figure 17-1. Optical Encoder Disk**

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 17-2.




**Legend:** N = lines per revolution

**Figure 17-2. 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



Copyright © 2024 Texas Instruments Incorporated

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 17-3. 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.

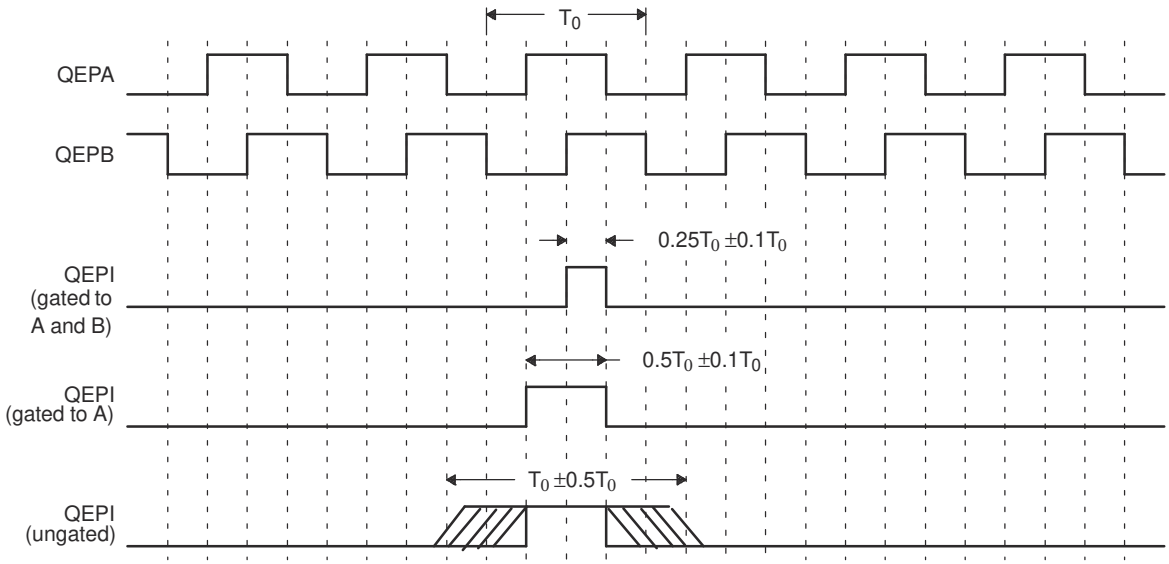



**Figure 17-3. Index Pulse Example**

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:

$$v(k) \approx \frac{x(k) - x(k-1)}{T} = \frac{\Delta X}{T} \quad (24)$$

$$v(k) \approx \frac{X}{t(k) - t(k-1)} = \frac{X}{\Delta T} \quad (25)$$

where:

- v(k) = Velocity at time instant k
- x(k) = Position at time instant k
- x(k-1) = Position at time instant k-1
- T = Fixed unit time or inverse of velocity calculation rate
- ΔX = Incremental position movement in unit time
- t(k) = Time instant "k"
- t(k-1) = Time instant "k-1"
- X = Fixed unit position
- ΔT = Incremental time elapsed for unit position movement



Copyright © 2024 Texas Instruments Incorporated

Equation 24 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 24 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 25 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 25 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 24. A combination of relatively large motor speeds and high sensor resolution makes the time interval ΔT small, and thus more greatly influenced by the timer resolution. This can introduce considerable error into high-speed estimates.

For systems with a large speed range (that is, speed estimation is needed at both low and high speeds), one approach is to use Equation 25 at low speed and have the DSP software switch over to Equation 24 when the motor speed rises above some specified threshold.

### *17.1.1 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*

## 17.2 Configuring Device Pins

The GPIO mux registers must be configured to connect this peripheral to the device pins. To avoid glitches on the pins, the GPyGMUX bits must be configured first (while keeping the corresponding GPyMUX bits at the default of zero), followed by writing the GPyMUX register to the desired value.

For proper 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.



Copyright © 2024 Texas Instruments Incorporated

## 17.3 Description

This section provides the eQEP inputs, memory map, and functional description.

### *17.3.1 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.



Copyright © 2024 Texas Instruments Incorporated

### 17.3.2 Functional Description

The eQEP peripheral contains the following major functional units (as shown in Figure 17-4):

- Programmable input qualification for each pin (part of the GPIO MUX)
- Quadrature decoder unit (QDU)
- Position counter and control unit for position measurement (PCCU)
- Quadrature edge-capture unit for low-speed measurement (QCAP)
- Unit time base for speed/frequency measurement (UTIME)
- Watchdog timer for detecting stalls (QWDOG)

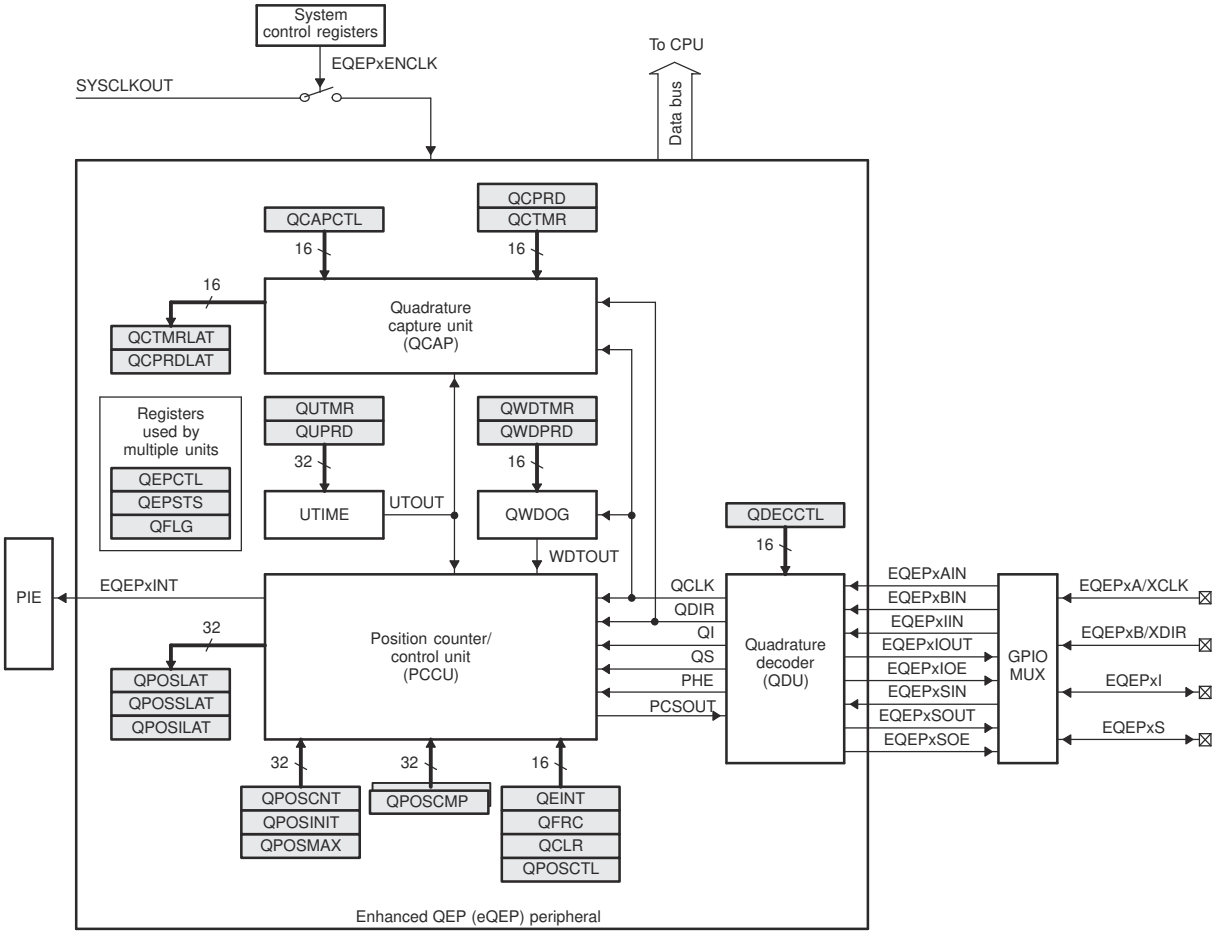



**Figure 17-4. Functional Block Diagram of the eQEP Peripheral**



Copyright © 2024 Texas Instruments Incorporated

### 17.3.3 eQEP Memory Map

Table 17-1 lists the registers with the memory locations, sizes, and reset values.

**Table 17-1. 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 0x3F | 31/0 | | |



Copyright © 2024 Texas Instruments Incorporated

## 17.4 Quadrature Decoder Unit (QDU)

Figure 17-5 shows a functional block diagram of the QDU.

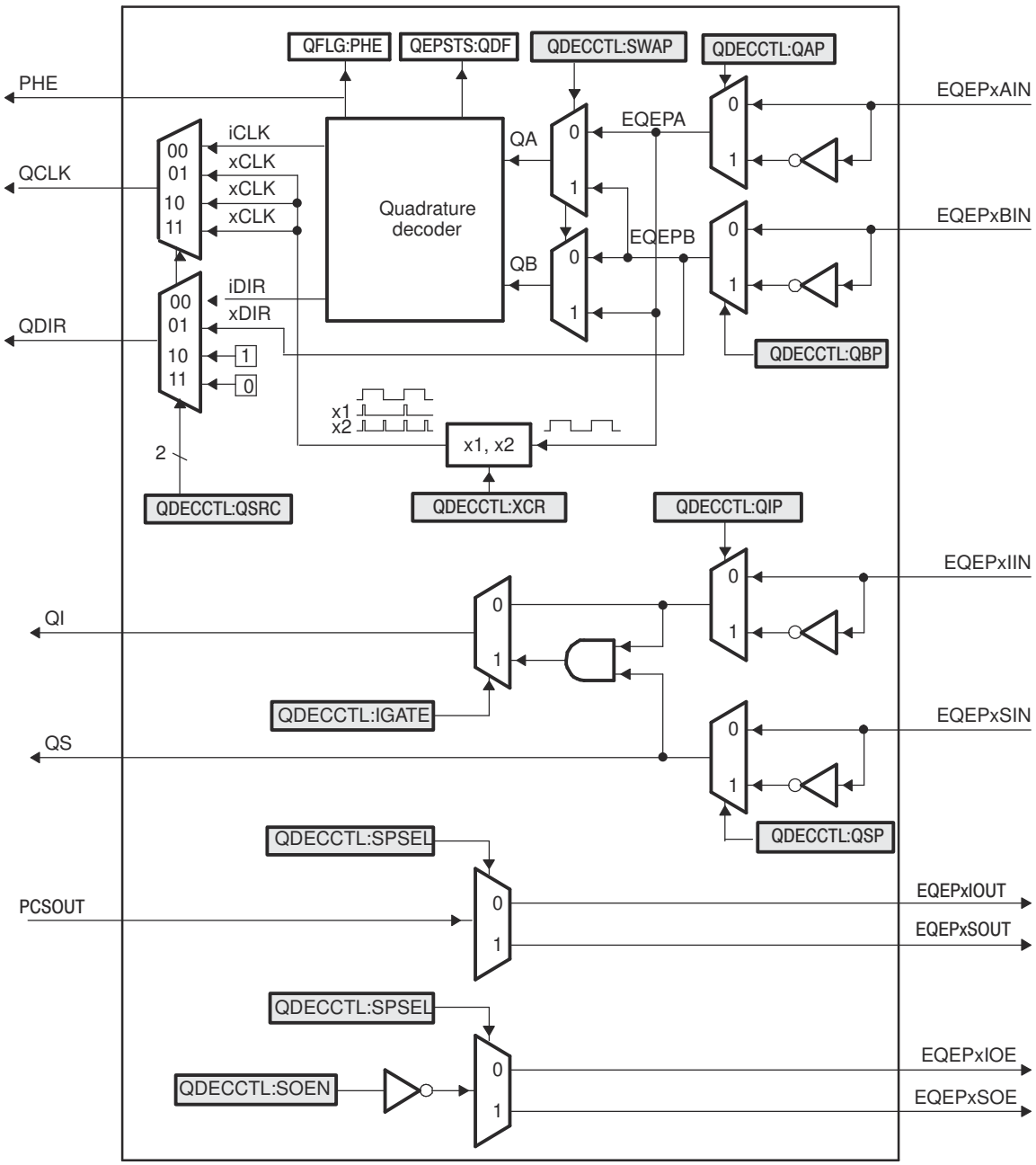



**Figure 17-5. Functional Block Diagram of Decoder Unit**

### *17.4.1 Position Counter Input Modes*

Clock and direction input to the position counter is selected using QDECCTL[QSRC] bits, based on interface input requirement as follows:

- Quadrature-count mode
- Direction-count mode
- UP-count mode
- DOWN-count mode



Copyright © 2024 Texas Instruments Incorporated

#### 17.4.1.1 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 17-2 and Figure 17-6 show the direction decoding logic in truth table and state machine form. Both edges of the QEPA and QEPB signals are sensed to generate count pulses for the position counter. Therefore, the frequency of the clock generated by the eQEP logic is four times that of each input sequence. Figure 17-7 shows the direction decoding and clock generation from the eQEP input signals.

**Table 17-2. Quadrature Decoder Truth Table**

| Previous Edge | Present Edge | QDIR | QPOSCNT |
|---|---|---|---|
| QA↑ | QB↑ | UP | Increment |
| | QB↓ | DOWN | Decrement |
| | QA↓ | TOGGLE | Increment or Decrement |
| QA↓ | QB↓ | UP | Increment |
| | QB↑ | DOWN | Decrement |
| | QA↑ | TOGGLE | Increment or Decrement |
| QB↑ | QA↑ | DOWN | Decrement |
| | QA↓ | UP | Increment |
| | QB↓ | TOGGLE | Increment or Decrement |
| QB↓ | QA↓ | DOWN | Decrement |
| | QA↑ | UP | Increment |
| | QB↑ | TOGGLE | Increment or Decrement |

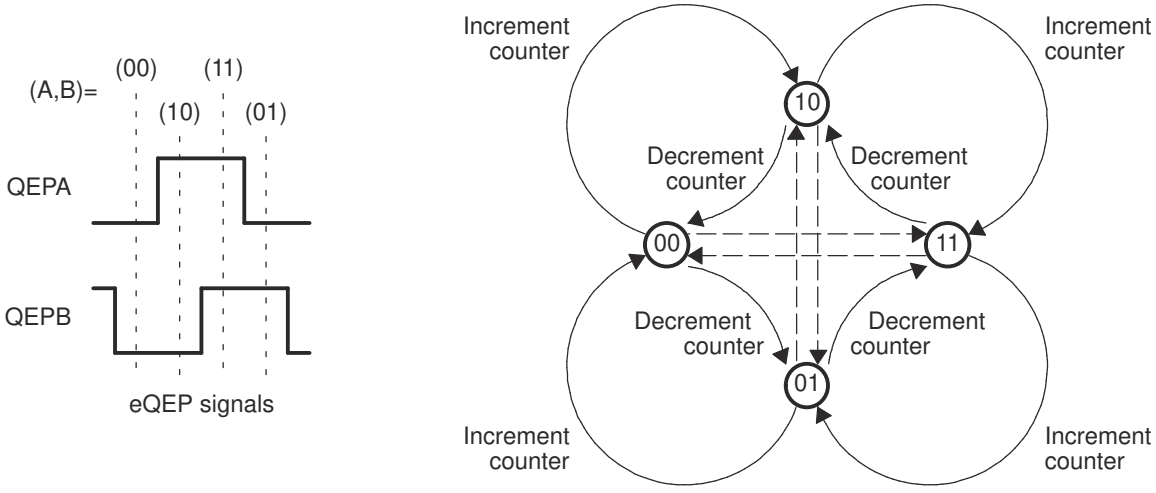



**Figure 17-6. Quadrature Decoder State Machine**



Copyright © 2024 Texas Instruments Incorporated

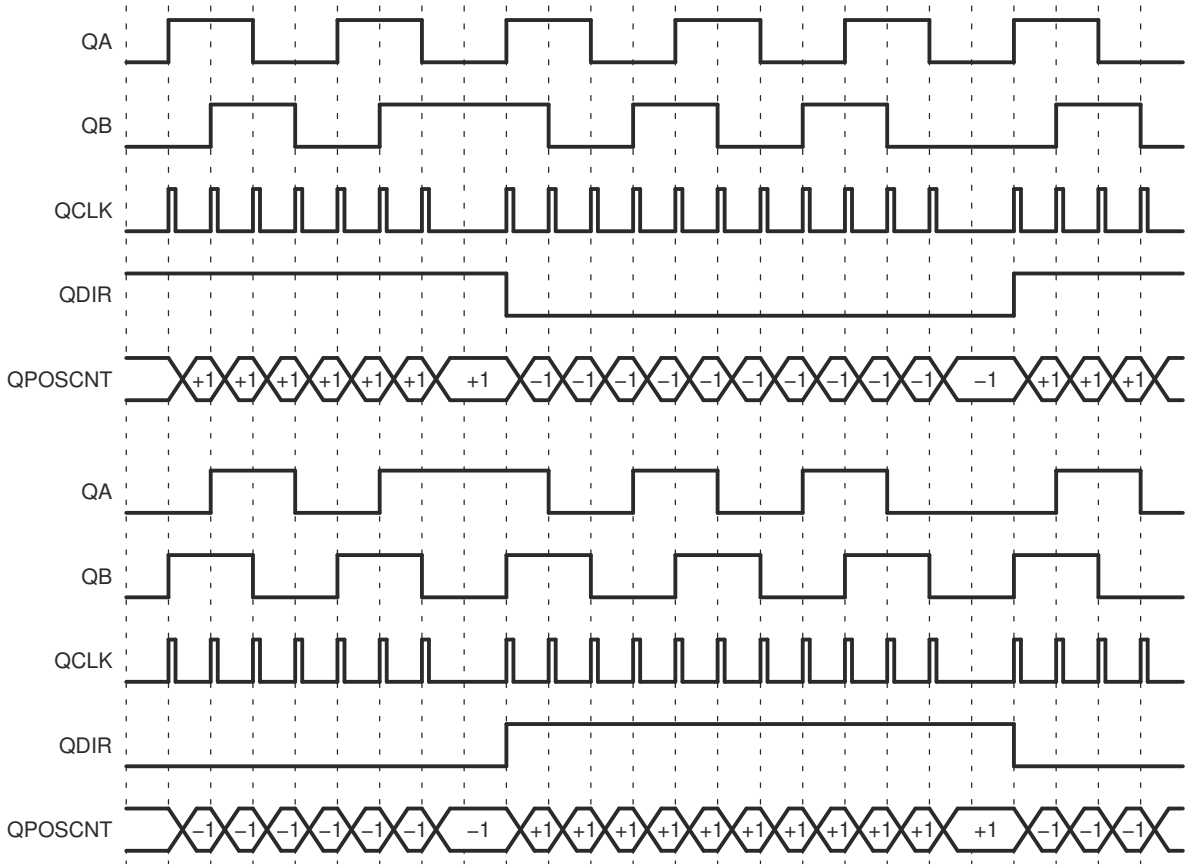



**Figure 17-7. Quadrature-clock and Direction Decoding**

**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 17-6 are invalid transitions that generate a phase error.

**Count Multiplication** The eQEP position counter provides 4x times the resolution of an input clock by generating a quadrature-clock (QCLK) on the rising/falling edges of both eQEP input clocks (QEPA and QEPB) as shown in Figure 17-7.

**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.

#### 17.4.1.2 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.



Copyright © 2024 Texas Instruments Incorporated

#### 17.4.1.3 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.

#### 17.4.1.4 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.

### *17.4.2 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.

### *17.4.3 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.

## 17.5 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.

### *17.5.1 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.



Copyright © 2024 Texas Instruments Incorporated

#### 17.5.1.1 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 QPOSMAX 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 17-8.

The position-counter value is latched to the QPOSILAT register and direction information is recorded in the QEPSTS[QDLF] bit on every index event marker. The position-counter error flag (QEPSTS[PCEF]) and error interrupt flag (QFLG[PCE]) are set if the latched value is not equal to 0 or QPOSMAX. The position-counter error flag (QEPSTS[PCEF]) is updated on every index event marker and an interrupt flag (QFLG[PCE]) 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 pcrm=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 IPOSLAT register on every index marker.

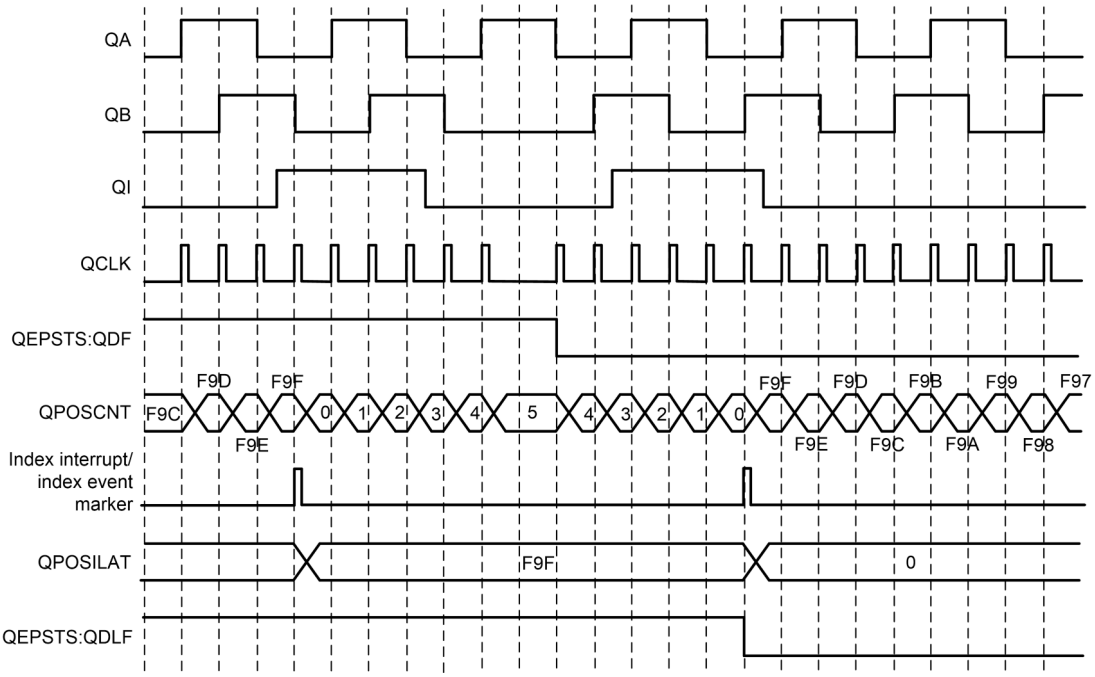



**Figure 17-8. Position Counter Reset by Index Pulse for 1000-Line Encoder (QPOSMAX = 3999 or 0xF9F)**

**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 QPOSMAX in the same SYSCLK cycle and does not wait for the next QCLK edge to occur.



Copyright © 2024 Texas Instruments Incorporated

#### 17.5.1.2 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 17-9 shows the position-counter reset operation in this mode.

The first index marker fields (QEPSTS[FIDF] and QEPSTS[FIMF]) are not applicable in this mode.

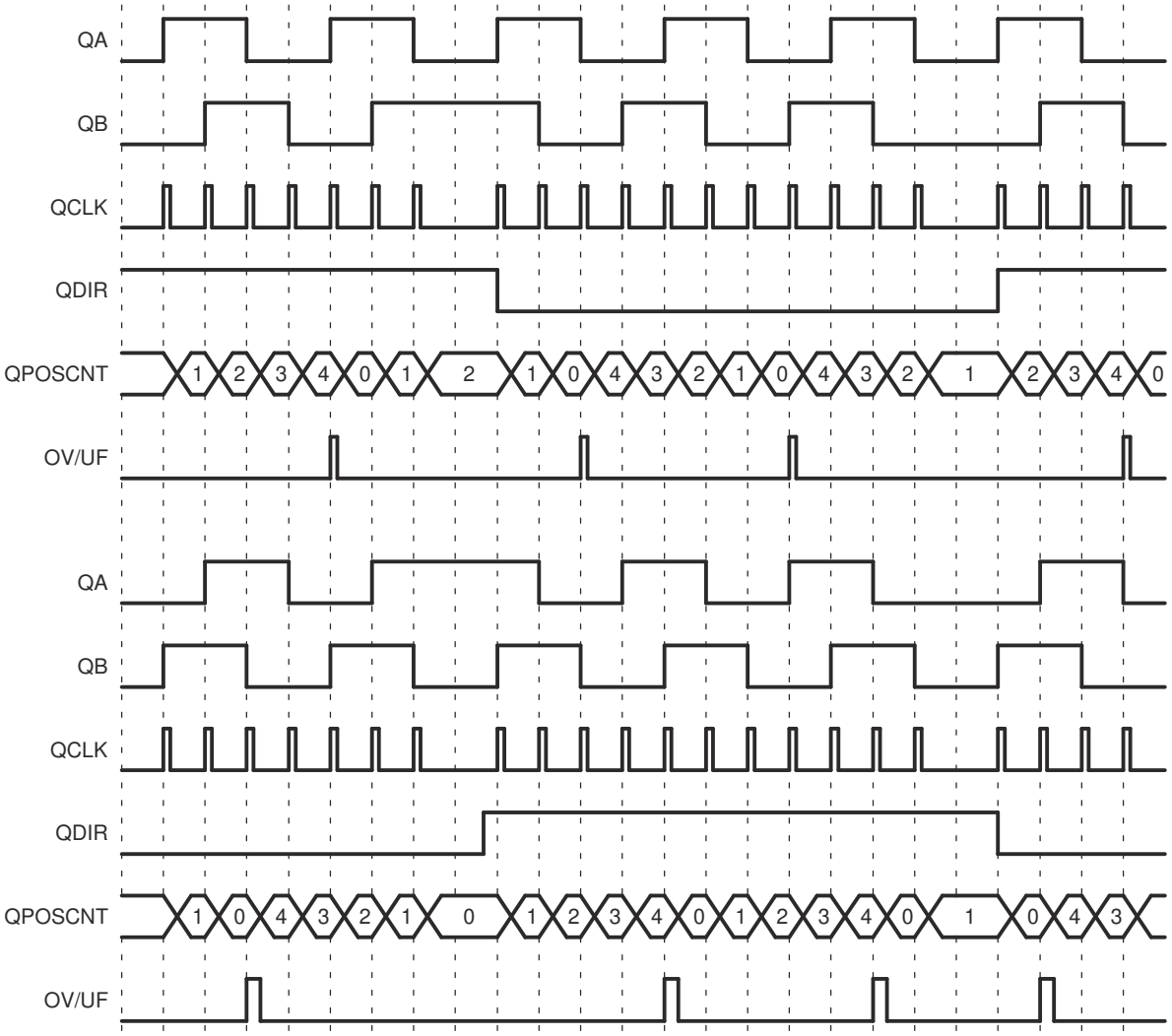



**Figure 17-9. Position Counter Underflow/Overflow (QPOSMAX = 4)**

#### 17.5.1.3 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 17.5.1.2.

The first index marker fields (QEPSTS[FIDF] and QEPSTS[FIMF]) are not applicable in this mode.



Copyright © 2024 Texas Instruments Incorporated

#### 17.5.1.4 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.

### *17.5.2 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.

#### 17.5.2.1 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.

| | |
|---|---|
| **Latch on Rising Edge (QEPCTL[IEL]=01)** | The position-counter value (QPOSCNT) is latched to the QPOSILAT register on every rising edge of an index input. |
| **Latch on Falling Edge (QEPCTL[IEL] = 10)** | The position-counter value (QPOSCNT) is latched to the QPOSILAT register on every falling edge of index input. |
| **Latch on Index Event Marker/Software Index Marker (QEPCTL[IEL] = 11** | The first index marker is defined as the quadrature edge following the first index edge. The eQEP peripheral records the occurrence of the first index marker (QEPSTS[FIMF]) and 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 17-10 shows the position counter latch using an index event marker.



Copyright © 2024 Texas Instruments Incorporated

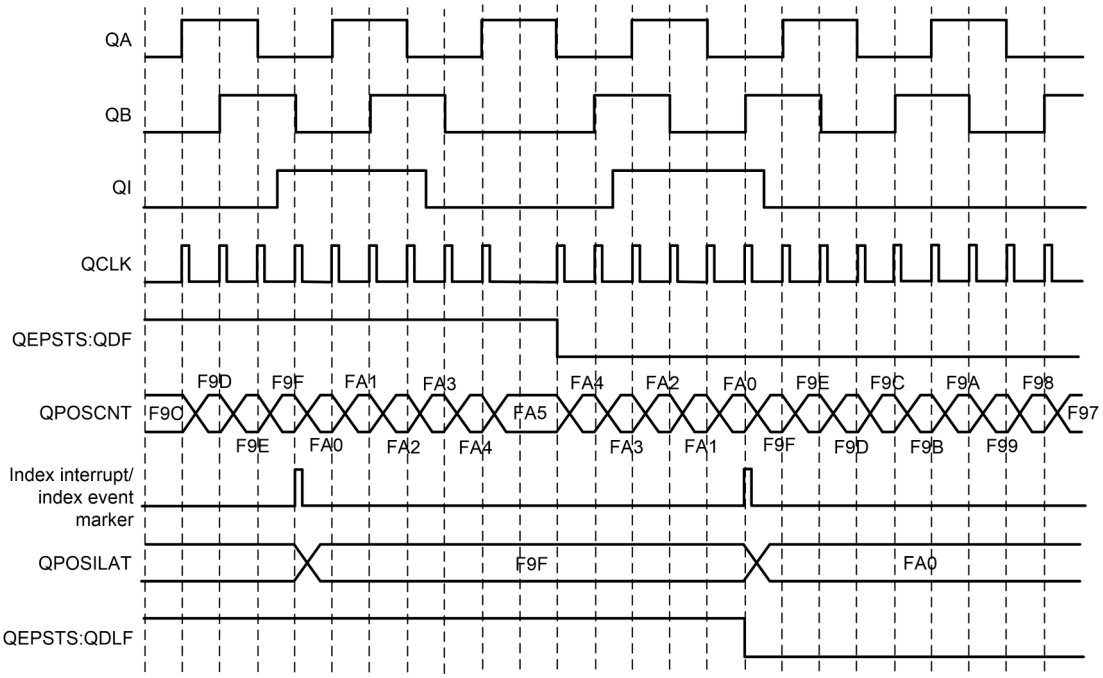



**Figure 17-10. Software Index Marker for 1000-line Encoder (QEPCTL[IEL] = 1)**

#### 17.5.2.2 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 17-11.

The strobe event latch interrupt flag (QFLG[SEL) is set when the position counter is latched to the QPOSSLAT register.

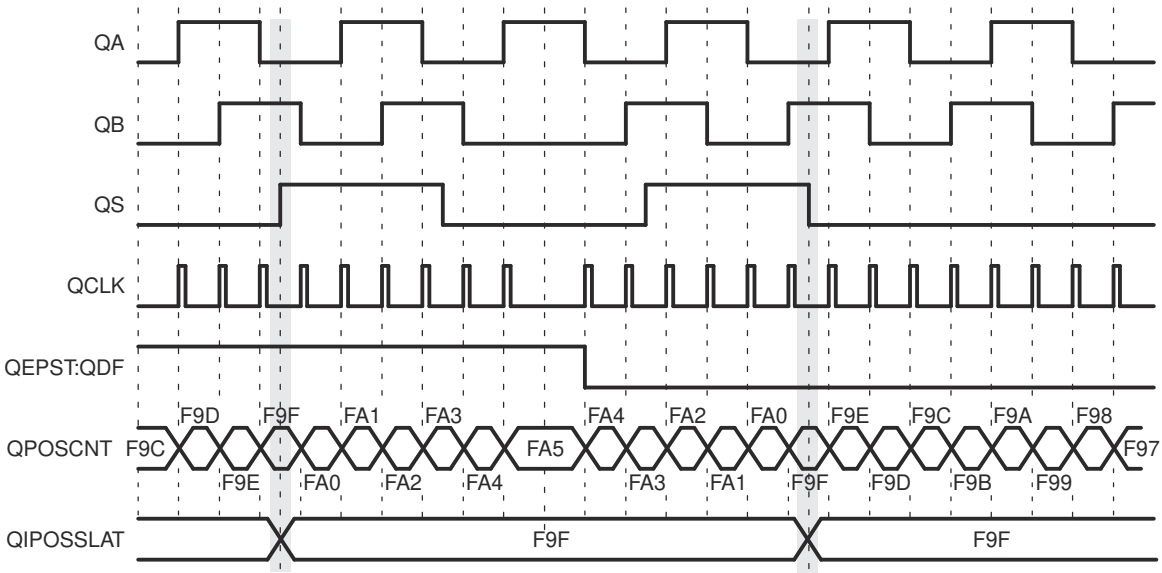



**Figure 17-11. Strobe Event Latch (QEPCTL[SEL] = 1)**



Copyright © 2024 Texas Instruments Incorporated

### 17.5.3 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.<br>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. |



Copyright © 2024 Texas Instruments Incorporated

### 17.5.4 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 17-12 shows a diagram. The position-compare (QPOSCMP) register is shadowed and shadow mode can be enabled or disabled using the QPOSCTL[PSSHDW] bit. If the shadow mode is not enabled, the CPU writes directly to the active position compare register.

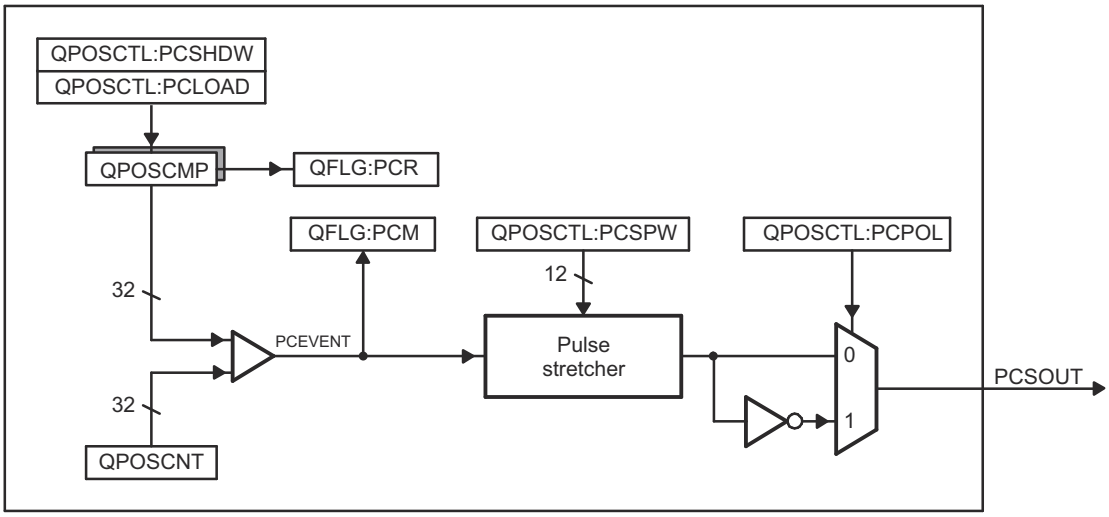



**Figure 17-12. eQEP Position-compare Unit**

In shadow mode, you can configure the position-compare unit (QPOSCTL[PCLOAD]) to load the shadow register value into the active register on the following events, and to generate the position-compare ready (QFLG[PCR]) interrupt after loading.

- Load on compare match
- Load on position-counter zero event

The position-compare match (QFLG[PCM]) is set when the position-counter value (QPOSCNT) matches with the active position-compare register (QPOSCMP) and the position-compare sync output of the programmable pulse width is generated on compare-match to trigger an external device.

For example, if QPOSCMP = 2, the position-compare unit generates a position-compare event on 1 to 2 transitions of the eQEP position counter for forward counting direction and on 3 to 2 transitions of the eQEP position counter for reverse counting direction (see Figure 17-13).

See the register section for the layout of the eQEP Position-Compare Control Register (QPOSCTL) and description of the QPOSCTL bit fields.



Copyright © 2024 Texas Instruments Incorporated

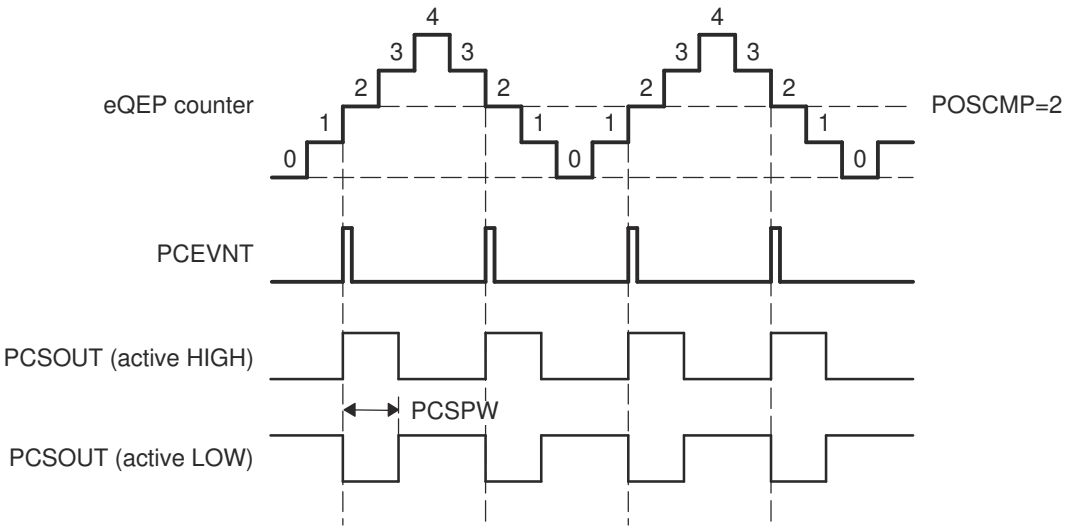



**Figure 17-13. eQEP Position-compare Event Generation Points**

The pulse stretcher logic in the position-compare unit generates a programmable position-compare sync pulse output on the position-compare match. In the event of a new position-compare match while a previous position-compare pulse is still active, then the pulse stretcher generates a pulse of specified duration from the new position-compare event as shown in Figure 17-14.

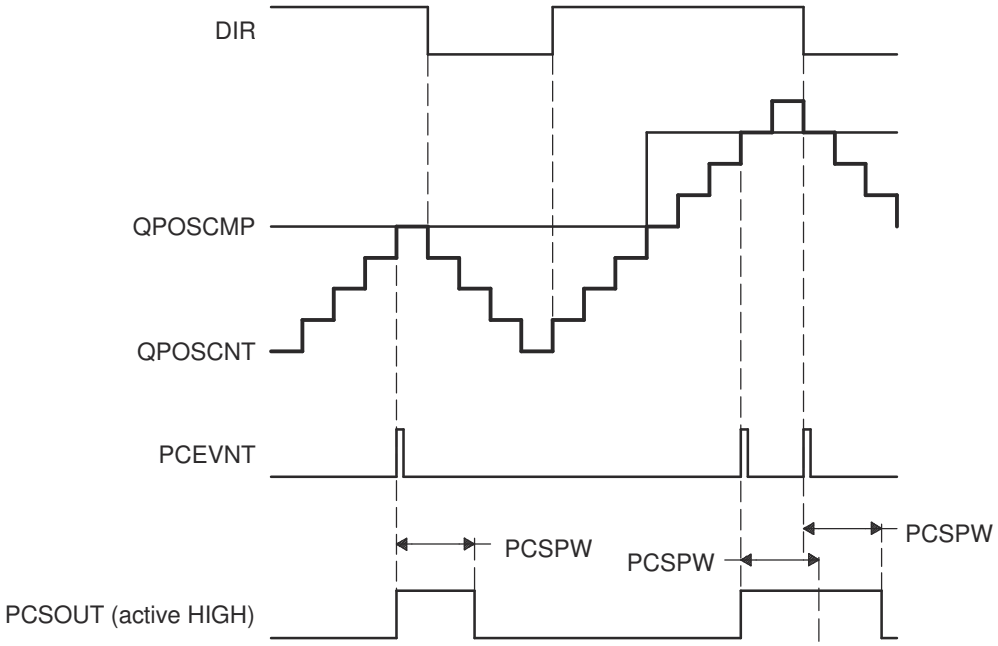



**Figure 17-14. eQEP Position-compare Sync Output Pulse Stretcher**



Copyright © 2024 Texas Instruments Incorporated

## 17.6 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 17-15. This feature is typically used for low-speed measurement using the following formula:

$$v(k) = \frac{X}{t(k) - t(k-1)} = \frac{X}{\Delta T} \quad (26)$$

where:

- X = Unit position is defined by integer multiple of quadrature edges (see Figure 17-16)
- ΔT = Elapsed time between unit position events
- v(k) = Velocity at time instant "k"

The eQEP capture timer (QCTMR) runs from prescaled SYSCLKOUT and the prescaler is programmed by the QCAPCTL[CCPS] bits. The capture timer (QCTMR) value is latched into the capture period register (QCPRD) on every unit position event and then the capture timer is reset, a flag is set in QEPSTS:UPEVNT to indicate that new value is latched into the QCPRD register. Software can check this status flag before reading the period register for low speed measurement, and clear the flag by writing 1.

Time measurement (ΔT) between unit position events 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 17-17 shows the capture unit operation along with the position counter.



Copyright © 2024 Texas Instruments Incorporated

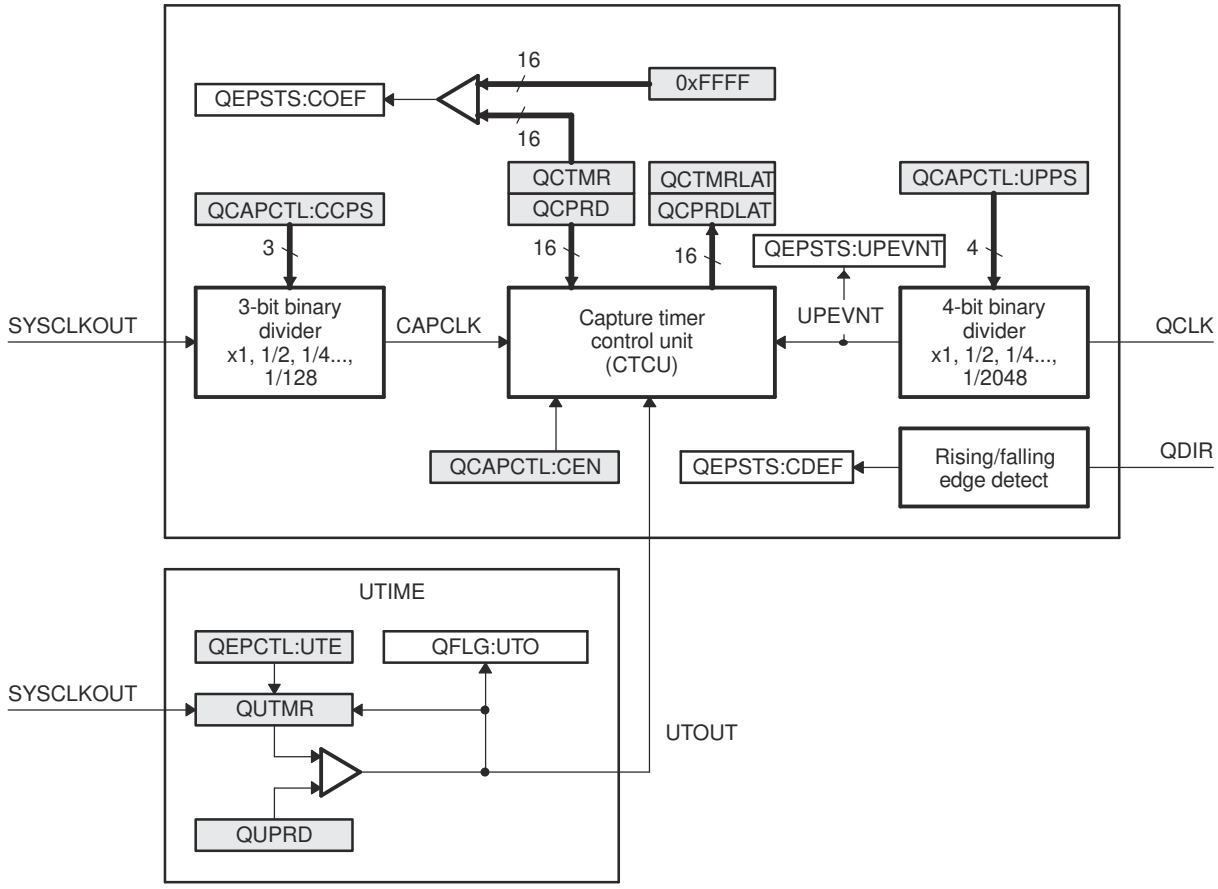


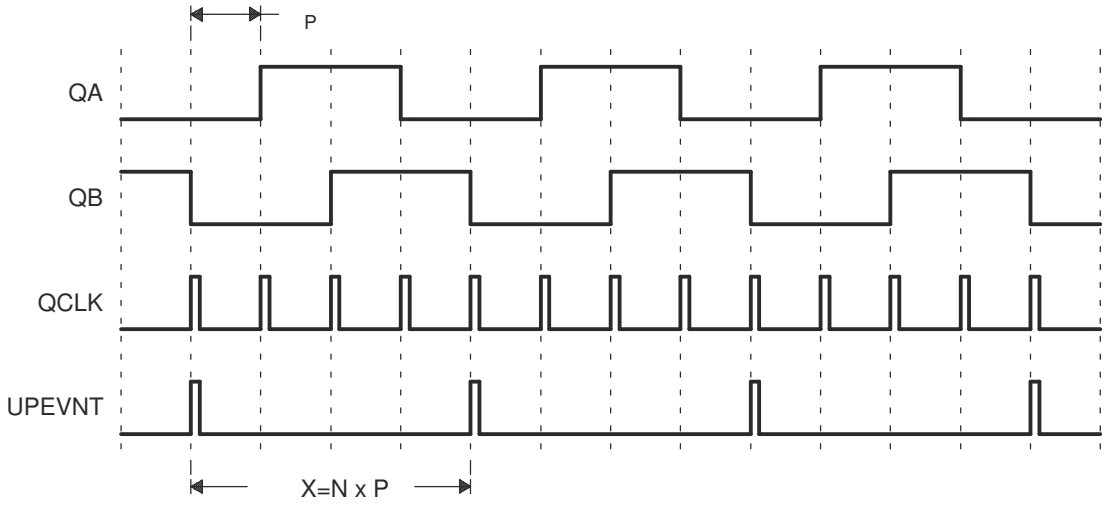

**Figure 17-15. eQEP Edge Capture Unit**

**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[CPPS] prescaler can be modified dynamically (such as switching CAPCLK prescaling mode from SYSCLK/4 to SYSCLK/8) only after the capture unit is disabled.



Copyright © 2024 Texas Instruments Incorporated




N = Number of quadrature periods selected using QCAPCTL[UPPS] bits

**Figure 17-16. Unit Position Event for Low Speed Measurement (QCAPCTL[UPPS] = 0010)**

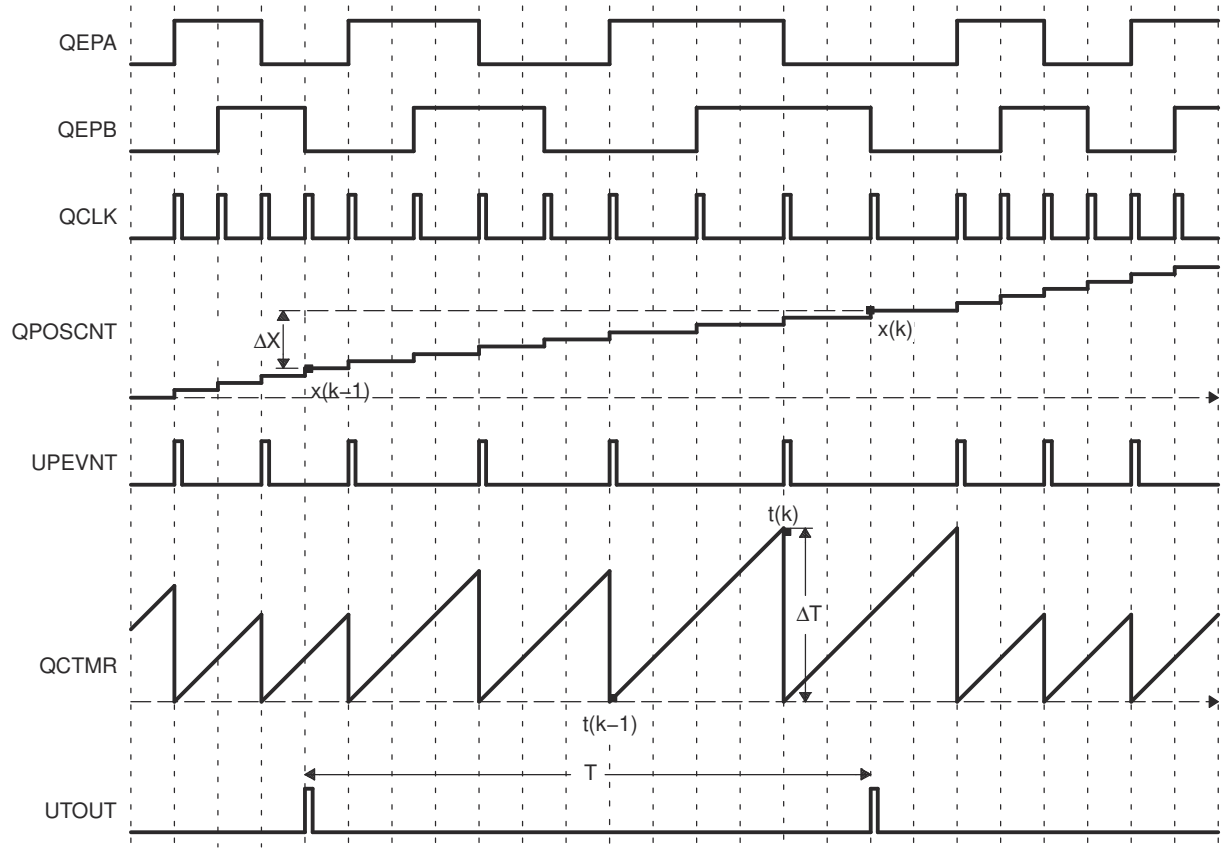



**Figure 17-17. eQEP Edge Capture Unit - Timing Details**



Copyright © 2024 Texas Instruments Incorporated

Velocity calculation equation:

$$v(k) = \frac{x(k) - x(k-1)}{T} = \frac{\Delta X}{T} \text{ o} \quad (27)$$

where:

- v(k) = Velocity at time instant k
- x(k) = Position at time instant k
- x(k-1) = Position at time instant k-1
- T = Fixed unit time or inverse of velocity calculation rate
- ΔX = Incremental position movement in unit time
- X = Fixed unit position
- ΔT = Incremental time elapsed for unit position movement
- t(k) = Time instant "k"
- t(k-1) = Time instant "k-1"

Unit time (T) and unit period (X) are configured using the QUPRD and QCAPCTL[UPPS] registers. Incremental position output and incremental time output is available in the QPOSLAT and QCPRDLAT registers.

| Parameter | Relevant Register to Configure or Read the Information |
|---|---|
| T | Unit Period Register (QUPRD) |
| ΔX | Incremental Position = QPOSLAT(k) - QPOSLAT(K-1) |
| X | Fixed-unit position defined by sensor resolution and QCAPCTL[UPPS] bits |
| ΔT | Capture Period Latch (QCPRDLAT) |



Copyright © 2024 Texas Instruments Incorporated

## 17.7 eQEP Watchdog

The eQEP peripheral contains a 16-bit watchdog timer (Figure 17-18) 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).

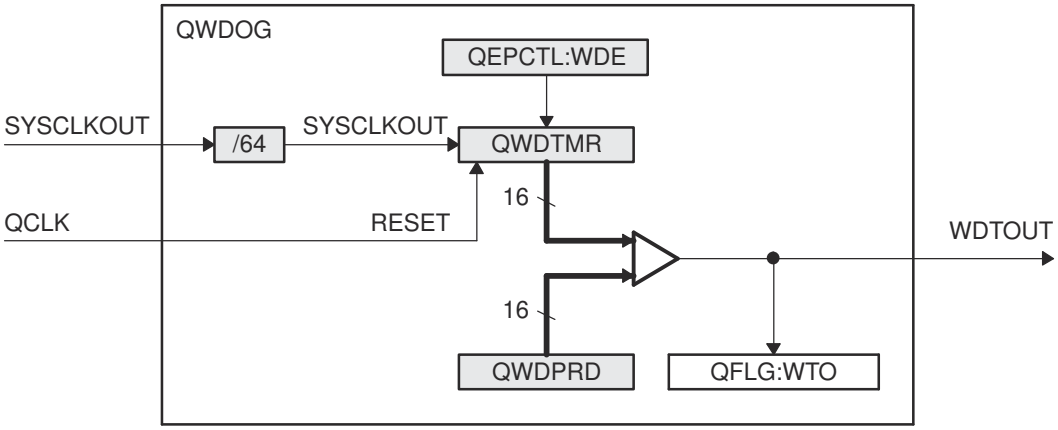



**Figure 17-18. eQEP Watchdog Timer**

## 17.8 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 17-19. 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 17.6.

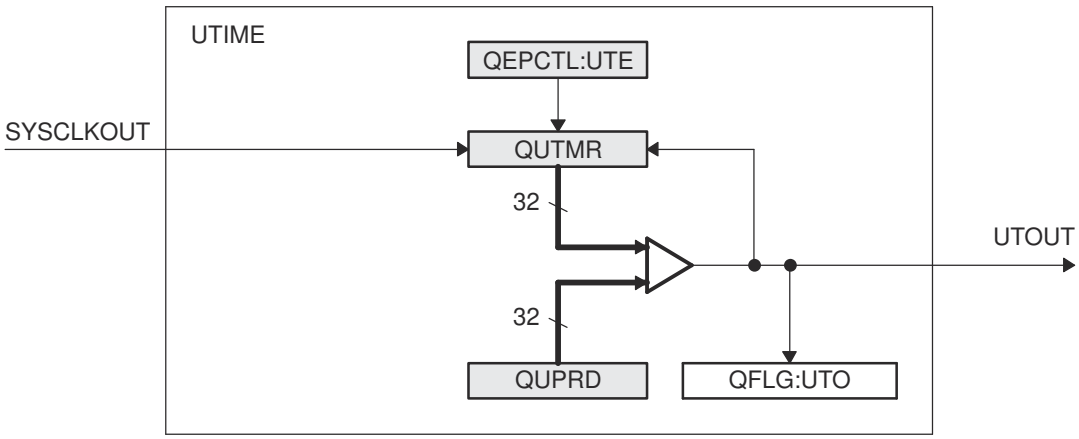



**Figure 17-19. eQEP Unit Timer Base**



Copyright © 2024 Texas Instruments Incorporated

## 17.9 eQEP Interrupt Structure

Figure 17-20 shows how the interrupt mechanism works in the eQEP module.

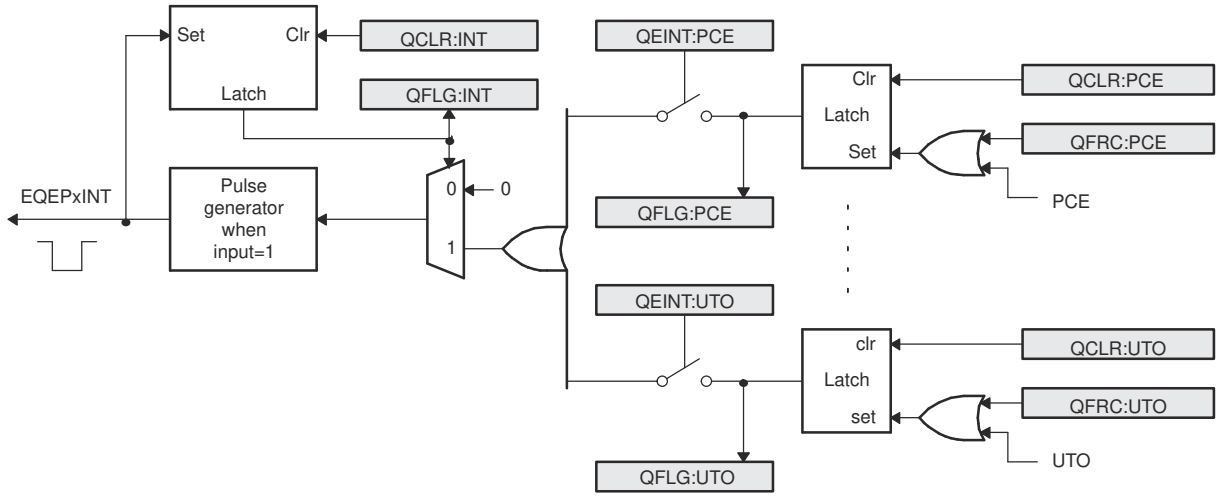



**Figure 17-20. eQEP Interrupt Generation**

Eleven interrupt events (PCE, PHE, QDC, WTO, PCU, PCO, PCR, PCM, SEL, IEL and UTO) can be generated. The interrupt control register (QEINT) is used to enable/disable individual interrupt event sources. The interrupt flag register (QFLG) indicates if any interrupt event has been latched and contains the global interrupt flag bit (INT).

An interrupt pulse is generated to 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 . You can force an interrupt event by way of the interrupt force register (QFRC), which is useful for test purposes.

## 17.10 eQEP Registers

This section describes the Enhanced Quadrature Encoder Pulse Registers.

### *17.10.1 eQEP Base Addresses*

**Table 17-3. eQEP Base Address Table**

| Device Registers | Register Name | Start Address | End Address |
|---|---|---|---|
| EQep1Regs | EQEP_REGS | 0x0000_5100 | 0x0000_513F |
| EQep2Regs | EQEP_REGS | 0x0000_5140 | 0x0000_517F |
| EQep3Regs | EQEP_REGS | 0x0000_5180 | 0x0000_51BF |



Copyright © 2024 Texas Instruments Incorporated

### 17.10.2 EQEP_REGS Registers

Table 17-4 lists the memory-mapped registers for the EQEP_REGS registers. All register offset addresses not listed in Table 17-4 should be considered as reserved locations and the register contents should not be modified.

**Table 17-4. EQEP_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | QPOSCNT | Position Counter | | Go |
| 2h | QPOSINIT | Position Counter Init | | Go |
| 4h | QPOSMAX | Maximum Position Count | | Go |
| 6h | QPOSCMP | Position Compare | | Go |
| 8h | QPOSILAT | Index Position Latch | | Go |
| Ah | QPOSSLAT | Strobe Position Latch | | Go |
| Ch | QPOSLAT | Position Latch | | Go |
| Eh | QUTMR | QEP Unit Timer | | Go |
| 10h | QUPRD | QEP Unit Period | | Go |
| 12h | QWDTMR | QEP Watchdog Timer | | Go |
| 13h | QWDPRD | QEP Watchdog Period | | Go |
| 14h | QDECCTL | Quadrature Decoder Control | | Go |
| 15h | QEPCTL | QEP Control | | Go |
| 16h | QCAPCTL | Qaudrature Capture Control | | Go |
| 17h | QPOSCTL | Position Compare Control | | Go |
| 18h | QEINT | QEP Interrupt Control | | Go |
| 19h | QFLG | QEP Interrupt Flag | | Go |
| 1Ah | QCLR | QEP Interrupt Clear | | Go |
| 1Bh | QFRC | QEP Interrupt Force | | Go |
| 1Ch | QEPSTS | QEP Status | | Go |
| 1Dh | QCTMR | QEP Capture Timer | | Go |
| 1Eh | QCPRD | QEP Capture Period | | Go |
| 1Fh | QCTMRLAT | QEP Capture Latch | | Go |
| 20h | QCPRDLAT | QEP Capture Period Latch | | Go |

Complex bit access types are encoded to fit into small table cells. Table 17-5 shows the codes that are used for access types in this section.

**Table 17-5. EQEP_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| R-0 | R<br>-0 | Read<br>Returns 0s |
| Write Type | | |
| W | W | Write |
| W1S | W<br>1S | Write<br>1 to set |
| Reset or Default Value | | |
| *-n* | | Value after reset or the default value |



Copyright © 2024 Texas Instruments Incorporated

### 17.10.2.1 QPOSCNT Register (Offset = 0h) [Reset = 00000000h]

QPOSCNT is shown in Figure 17-21 and described in Table 17-6.

Return to the Summary Table.

Position Counter

**Figure 17-21. QPOSCNT Register**

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

**Table 17-6. QPOSCNT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | QPOSCNT | R/W | 0h | Position Counter<br>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.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 17.10.2.2 QPOSINIT Register (Offset = 2h) [Reset = 00000000h]

QPOSINIT is shown in Figure 17-22 and described in Table 17-7.

Return to the Summary Table.

Position Counter Init

**Figure 17-22. QPOSINIT Register**

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

**Table 17-7. QPOSINIT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | QPOSINIT | R/W | 0h | Position Counter Init<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 17.10.2.3 QPOSMAX Register (Offset = 4h) [Reset = 00000000h]

QPOSMAX is shown in Figure 17-23 and described in Table 17-8.

Return to the Summary Table.

Maximum Position Count

**Figure 17-23. QPOSMAX Register**

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

**Table 17-8. QPOSMAX Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | QPOSMAX | R/W | 0h | Maximum Position Count<br>This register contains the maximum position counter value. Writes to this register should always be full 32-bit writes.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 17.10.2.4 QPOSCMP Register (Offset = 6h) [Reset = 00000000h]

QPOSCMP is shown in Figure 17-24 and described in Table 17-9.

Return to the Summary Table.

Position Compare

**Figure 17-24. QPOSCMP Register**

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

**Table 17-9. QPOSCMP Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | QPOSCMP | R/W | 0h | Position Compare<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 17.10.2.5 QPOSILAT Register (Offset = 8h) [Reset = 00000000h]

QPOSILAT is shown in Figure 17-25 and described in Table 17-10.

Return to the Summary Table.

Index Position Latch

**Figure 17-25. QPOSILAT Register**

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

**Table 17-10. QPOSILAT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | QPOSILAT | R | 0h | Index Position Latch<br>The position-counter value is latched into this register on an index event as defined by the QEPCTL[IEL] bits.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 17.10.2.6 QPOSSLAT Register (Offset = Ah) [Reset = 00000000h]

QPOSSLAT is shown in Figure 17-26 and described in Table 17-11.

Return to the Summary Table.

Strobe Position Latch

**Figure 17-26. QPOSSLAT Register**

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

**Table 17-11. QPOSSLAT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | QPOSSLAT | R | 0h | Strobe Position Latch<br>The position-counter value is latched into this register on a strobe event as defined by the QEPCTL[SEL] bits.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 17.10.2.7 QPOSLAT Register (Offset = Ch) [Reset = 00000000h]

QPOSLAT is shown in Figure 17-27 and described in Table 17-12.

Return to the Summary Table.

Position Latch

**Figure 17-27. QPOSLAT Register**

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

**Table 17-12. QPOSLAT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | QPOSLAT | R | 0h | Position Latch<br>The position-counter value is latched into this register on a unit time out event.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 17.10.2.8 QUTMR Register (Offset = Eh) [Reset = 00000000h]

QUTMR is shown in Figure 17-28 and described in Table 17-13.

Return to the Summary Table.

QEP Unit Timer

**Figure 17-28. QUTMR Register**

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

**Table 17-13. QUTMR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | QUTMR | R/W | 0h | QEP Unit Timer<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 17.10.2.9 QUPRD Register (Offset = 10h) [Reset = 00000000h]

QUPRD is shown in Figure 17-29 and described in Table 17-14.

Return to the Summary Table.

QEP Unit Period

**Figure 17-29. QUPRD Register**

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

**Table 17-14. QUPRD Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | QUPRD | R/W | 0h | QEP Unit Period<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 17.10.2.10 QWDTMR Register (Offset = 12h) [Reset = 0000h]

QWDTMR is shown in Figure 17-30 and described in Table 17-15.

Return to the Summary Table.

QEP Watchdog Timer

**Figure 17-30. 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 17-15. QWDTMR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | QWDTMR | R/W | 0h | QEP Watchdog Timer<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 17.10.2.11 QWDPRD Register (Offset = 13h) [Reset = 0000h]

QWDPRD is shown in Figure 17-31 and described in Table 17-16.

Return to the Summary Table.

QEP Watchdog Period

**Figure 17-31. 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 17-16. QWDPRD Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | QWDPRD | R/W | 0h | QEP Watchdog Period<br>This register contains the time-out count for the eQEP peripheral watch dog timer.<br>When the watchdog timer value matches the watchdog period value, a watchdog timeout interrupt is generated.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 17.10.2.12 QDECCTL Register (Offset = 14h) [Reset = 0000h]

QDECCTL is shown in Figure 17-32 and described in Table 17-17.

Return to the Summary Table.

Quadrature Decoder Control

**Figure 17-32. QDECCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| QSRC | | SOEN | SPSEL | XCR | SWAP | IGATE | QAP |
| R/W-0h | | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| QBP | QIP | QSP | RESERVED | | | | |
| R/W-0h | R/W-0h | R/W-0h | R-0h | | | | |

**Table 17-17. QDECCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-14 | QSRC | R/W | 0h | Position-counter source selection<br>Reset type: SYSRSn<br>0h (R/W) = Quadrature count mode (QCLK = iCLK, QDIR = iDIR)<br>1h (R/W) = Direction-count mode (QCLK = xCLK, QDIR = xDIR)<br>2h (R/W) = UP count mode for frequency measurement (QCLK = xCLK, QDIR = 1)<br>3h (R/W) = DOWN count mode for frequency measurement (QCLK = xCLK, QDIR = 0) |
| 13 | SOEN | R/W | 0h | Sync output-enable<br>Reset type: SYSRSn<br>0h (R/W) = Disable position-compare sync output<br>1h (R/W) = Enable position-compare sync output |
| 12 | SPSEL | R/W | 0h | Sync output pin selection<br>Reset type: SYSRSn<br>0h (R/W) = Index pin is used for sync output<br>1h (R/W) = Strobe pin is used for sync output |
| 11 | XCR | R/W | 0h | External Clock Rate<br>Reset type: SYSRSn<br>0h (R/W) = 2x resolution: Count the rising/falling edge<br>1h (R/W) = 1x resolution: Count the rising edge only |
| 10 | SWAP | R/W | 0h | CLK/DIR Signal Source for Position Counter<br>Reset type: SYSRSn<br>0h (R/W) = Quadrature-clock inputs are not swapped<br>1h (R/W) = Quadrature-clock inputs are swapped |
| 9 | IGATE | R/W | 0h | Index pulse gating option<br>Reset type: SYSRSn<br>0h (R/W) = Disable gating of Index pulse<br>1h (R/W) = Gate the index pin with strobe |
| 8 | QAP | R/W | 0h | QEPA input polarity<br>Reset type: SYSRSn<br>0h (R/W) = No effect<br>1h (R/W) = Negates QEPA input |
| 7 | QBP | R/W | 0h | QEPB input polarity<br>Reset type: SYSRSn<br>0h (R/W) = No effect<br>1h (R/W) = Negates QEPB input |
| 6 | QIP | R/W | 0h | QEPI input polarity<br>Reset type: SYSRSn<br>0h (R/W) = No effect<br>1h (R/W) = Negates QEPI input |



Copyright © 2024 Texas Instruments Incorporated

**Table 17-17. QDECCTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 5 | QSP | R/W | 0h | QEPS input polarity<br>Reset type: SYSRSn<br>0h (R/W) = No effect<br>1h (R/W) = Negates QEPS input |
| 4-0 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 17.10.2.13 QEPCTL Register (Offset = 15h) [Reset = 0000h]

QEPCTL is shown in Figure 17-33 and described in Table 17-18.

Return to the Summary Table.

QEP Control

**Figure 17-33. QEPCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| FREE_SOFT | | PCRM | | SEI | | IEI | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SWI | SEL | IEL | | QPEN | QCLM | UTE | WDE |
| R/W-0h | R/W-0h | R/W-0h | | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 17-18. QEPCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-14 | FREE_SOFT | R/W | 0h | Emulation mode<br>Reset type: SYSRSn<br>0h (R/W) = QPOSCNT behavior<br>Position counter stops immediately on emulation suspend<br>0h (R/W) = QWDTMR behavior<br>Watchdog counter stops immediately<br>0h (R/W) = QUTMR behavior<br>Unit timer stops immediately<br>0h (R/W) = QCTMR behavior<br>Capture Timer stops immediately<br>1h (R/W) = QPOSCNT behavior<br>Position counter continues to count until the rollover<br>1h (R/W) = QWDTMR behavior<br>Watchdog counter counts until WD period match roll over<br>1h (R/W) = QUTMR behavior<br>Unit timer counts until period rollover<br>1h (R/W) = QCTMR behavior<br>Capture Timer counts until next unit period event<br>2h (R/W) = QPOSCNT behavior<br>Position counter is unaffected by emulation suspend<br>2h (R/W) = QWDTMR behavior<br>Watchdog counter is unaffected by emulation suspend<br>2h (R/W) = QUTMR behavior<br>Unit timer is unaffected by emulation suspend<br>2h (R/W) = QCTMR behavior<br>Capture Timer is unaffected by emulation suspend<br>3h (R/W) = Same as FREE_SOFT_2 |
| 13-12 | PCRM | R/W | 0h | Postion counter reset<br>Reset type: SYSRSn<br>0h (R/W) = Position counter reset on an index event<br>1h (R/W) = Position counter reset on the maximum position<br>2h (R/W) = Position counter reset on the first index event<br>3h (R/W) = Position counter reset on a unit time event |
| 11-10 | SEI | R/W | 0h | Strobe event initialization of position counter<br>Reset type: SYSRSn<br>0h (R/W) = Does nothing (action disabled)<br>1h (R/W) = Does nothing (action disabled)<br>2h (R/W) = Initializes the position counter on rising edge of the QEPS signal<br>3h (R/W) = Clockwise Direction:<br>Initializes the position counter on the rising edge of QEPS strobe<br>Counter Clockwise Direction:<br>Initializes the position counter on the falling edge of QEPS strobe |



Copyright © 2024 Texas Instruments Incorporated

**Table 17-18. QEPCTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 9-8 | IEI | R/W | 0h | Index event init of position count<br>Reset type: SYSRSn<br>0h (R/W) = Do nothing (action disabled)<br>1h (R/W) = Do nothing (action disabled)<br>2h (R/W) = Initializes the position counter on the rising edge of the QEPI signal (QPOSCNT = QPOSINIT)<br>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<br>Reset type: SYSRSn<br>0h (R/W) = Do nothing (action disabled)<br>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<br>Reset type: SYSRSn<br>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<br>1h (R/W) = Clockwise Direction:<br>Position counter is latched on rising edge of QEPS strobe<br>Counter Clockwise Direction:<br>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)<br>Reset type: SYSRSn<br>0h (R/W) = Reserved<br>1h (R/W) = Latches position counter on rising edge of the index signal<br>2h (R/W) = Latches position counter on falling edge of the index signal<br>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<br>Reset type: SYSRSn<br>0h (R/W) = Reset the eQEP peripheral internal operating flags/read-only registers. Control/configuration registers are not disturbed by a software reset.<br>When QPEN is disabled, some flags in the QFLG register do not get reset or cleared and show the actual state of that flag.<br>1h (R/W) = eQEP position counter is enabled |
| 2 | QCLM | R/W | 0h | QEP capture latch mode<br>Reset type: SYSRSn<br>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.<br>1h (R/W) = Latch on unit time out. Position counter, capture timer and capture period values are latched into QPOSLAT, QCTMRLAT and QCPRDLAT registers on unit time out. |
| 1 | UTE | R/W | 0h | QEP unit timer enable<br>Reset type: SYSRSn<br>0h (R/W) = Disable eQEP unit timer<br>1h (R/W) = Enable unit timer |
| 0 | WDE | R/W | 0h | QEP watchdog enable<br>Reset type: SYSRSn<br>0h (R/W) = Disable the eQEP watchdog timer<br>1h (R/W) = Enable the eQEP watchdog timer |



Copyright © 2024 Texas Instruments Incorporated

### 17.10.2.14 QCAPCTL Register (Offset = 16h) [Reset = 0000h]

QCAPCTL is shown in Figure 17-34 and described in Table 17-19.

Return to the Summary Table.

Qaudrature Capture Control

**Figure 17-34. QCAPCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| CEN | RESERVED | | | | | | |
| R/W-0h | R-0h | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | CCPS | | | UPPS | | | |
| R-0h | R/W-0h | | | R/W-0h | | | |

**Table 17-19. QCAPCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | CEN | R/W | 0h | Enable eQEP capture<br>Reset type: SYSRSn<br>0h (R/W) = eQEP capture unit is disabled<br>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<br>Reset type: SYSRSn<br>0h (R/W) = CAPCLK = SYSCLKOUT/1<br>1h (R/W) = CAPCLK = SYSCLKOUT/2<br>2h (R/W) = CAPCLK = SYSCLKOUT/4<br>3h (R/W) = CAPCLK = SYSCLKOUT/8<br>4h (R/W) = CAPCLK = SYSCLKOUT/16<br>5h (R/W) = CAPCLK = SYSCLKOUT/32<br>6h (R/W) = CAPCLK = SYSCLKOUT/64<br>7h (R/W) = CAPCLK = SYSCLKOUT/128 |
| 3-0 | UPPS | R/W | 0h | Unit position event prescaler<br>Reset type: SYSRSn<br>0h (R/W) = UPEVNT = QCLK/1<br>1h (R/W) = UPEVNT = QCLK/2<br>2h (R/W) = UPEVNT = QCLK/4<br>3h (R/W) = UPEVNT = QCLK/8<br>4h (R/W) = UPEVNT = QCLK/16<br>5h (R/W) = UPEVNT = QCLK/32<br>6h (R/W) = UPEVNT = QCLK/64<br>7h (R/W) = UPEVNT = QCLK/128<br>8h (R/W) = UPEVNT = QCLK/256<br>9h (R/W) = UPEVNT = QCLK/512<br>Ah (R/W) = UPEVNT = QCLK/1024<br>Bh (R/W) = UPEVNT = QCLK/2048<br>Ch (R/W) = Reserved<br>Dh (R/W) = Reserved<br>Eh (R/W) = Reserved<br>Fh (R/W) = Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 17.10.2.15 QPOSCTL Register (Offset = 17h) [Reset = 0000h]

QPOSCTL is shown in Figure 17-35 and described in Table 17-20.

Return to the Summary Table.

Position Compare Control

**Figure 17-35. QPOSCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| PCSHDW | PCLOAD | PCPOL | PCE | PCSPW | | | |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PCSPW | | | | | | | |
| R/W-0h | | | | | | | |

**Table 17-20. QPOSCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | PCSHDW | R/W | 0h | Position compare of shadow enable<br>Reset type: SYSRSn<br>0h (R/W) = Shadow disabled, load Immediate<br>1h (R/W) = Shadow enabled |
| 14 | PCLOAD | R/W | 0h | Position compare of shadow load<br>Reset type: SYSRSn<br>0h (R/W) = Load on QPOSCNT = 0<br>1h (R/W) = Load when QPOSCNT = QPOSCMP |
| 13 | PCPOL | R/W | 0h | Polarity of sync output<br>Reset type: SYSRSn<br>0h (R/W) = Active HIGH pulse output<br>1h (R/W) = Active LOW pulse output |
| 12 | PCE | R/W | 0h | Position compare enable/disable<br>Reset type: SYSRSn<br>0h (R/W) = Disable position compare unit<br>1h (R/W) = Enable position compare unit |
| 11-0 | PCSPW | R/W | 0h | Select-position-compare sync output pulse width<br>Reset type: SYSRSn<br>0h (R/W) = 1 * 4 * SYSCLKOUT cycles<br>1h (R/W) = 2 * 4 * SYSCLKOUT cycles<br>FFFh (R/W) = 4096 * 4 * SYSCLKOUT cycles |



Copyright © 2024 Texas Instruments Incorporated

### 17.10.2.16 QEINT Register (Offset = 18h) [Reset = 0000h]

QEINT is shown in Figure 17-36 and described in Table 17-21.

Return to the Summary Table.

QEP Interrupt Control

**Figure 17-36. QEINT Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | UTO | IEL | SEL | PCM |
| R-0h | | | | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PCR | PCO | PCU | WTO | QDC | QPE | PCE | RESERVED |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R-0h |

**Table 17-21. QEINT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | RESERVED | R | 0h | Reserved |
| 11 | UTO | R/W | 0h | Unit time out interrupt enable<br>Reset type: SYSRSn<br>0h (R/W) = Interrupt is disabled<br>1h (R/W) = Interrupt is enabled |
| 10 | IEL | R/W | 0h | Index event latch interrupt enable<br>Reset type: SYSRSn<br>0h (R/W) = Interrupt is disabled<br>1h (R/W) = Interrupt is enabled |
| 9 | SEL | R/W | 0h | Strobe event latch interrupt enable<br>Reset type: SYSRSn<br>0h (R/W) = Interrupt is disabled<br>1h (R/W) = Interrupt is enabled |
| 8 | PCM | R/W | 0h | Position-compare match interrupt enable<br>Reset type: SYSRSn<br>0h (R/W) = Interrupt is disabled<br>1h (R/W) = Interrupt is enabled |
| 7 | PCR | R/W | 0h | Position-compare ready interrupt enable<br>Reset type: SYSRSn<br>0h (R/W) = Interrupt is disabled<br>1h (R/W) = Interrupt is enabled |
| 6 | PCO | R/W | 0h | Position counter overflow interrupt enable<br>Reset type: SYSRSn<br>0h (R/W) = Interrupt is disabled<br>1h (R/W) = Interrupt is enabled |
| 5 | PCU | R/W | 0h | Position counter underflow interrupt enable<br>Reset type: SYSRSn<br>0h (R/W) = Interrupt is disabled<br>1h (R/W) = Interrupt is enabled |
| 4 | WTO | R/W | 0h | Watchdog time out interrupt enable<br>Reset type: SYSRSn<br>0h (R/W) = Interrupt is disabled<br>1h (R/W) = Interrupt is enabled |
| 3 | QDC | R/W | 0h | Quadrature direction change interrupt enable<br>Reset type: SYSRSn<br>0h (R/W) = Interrupt is disabled<br>1h (R/W) = Interrupt is enabled |



Copyright © 2024 Texas Instruments Incorporated

**Table 17-21. QEINT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 2 | QPE | R/W | 0h | Quadrature phase error interrupt enable<br>Reset type: SYSRSn<br>0h (R/W) = Interrupt is disabled<br>1h (R/W) = Interrupt is enabled |
| 1 | PCE | R/W | 0h | Position counter error interrupt enable<br>Reset type: SYSRSn<br>0h (R/W) = Interrupt is disabled<br>1h (R/W) = Interrupt is enabled |
| 0 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 17.10.2.17 QFLG Register (Offset = 19h) [Reset = 0000h]

QFLG is shown in Figure 17-37 and described in Table 17-22.

Return to the Summary Table.

QEP Interrupt Flag

**Figure 17-37. QFLG Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | UTO | IEL | SEL | PCM |
| R-0h | | | | R-0h | R-0h | R-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PCR | PCO | PCU | WTO | QDC | PHE | PCE | INT |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 17-22. QFLG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | RESERVED | R | 0h | Reserved |
| 11 | UTO | R | 0h | Unit time out interrupt flag<br>Reset type: SYSRSn<br>0h (R/W) = No interrupt generated<br>1h (R/W) = Interrupt was generated |
| 10 | IEL | R | 0h | Index event latch interrupt flag<br>Reset type: SYSRSn<br>0h (R/W) = No interrupt generated<br>1h (R/W) = Interrupt was generated |
| 9 | SEL | R | 0h | Strobe event latch interrupt flag<br>Reset type: SYSRSn<br>0h (R/W) = No interrupt generated<br>1h (R/W) = Interrupt was generated |
| 8 | PCM | R | 0h | eQEP compare match event interrupt flag<br>Reset type: SYSRSn<br>0h (R/W) = No interrupt generated<br>1h (R/W) = Interrupt was generated |
| 7 | PCR | R | 0h | Position-compare ready interrupt flag<br>Reset type: SYSRSn<br>0h (R/W) = No interrupt generated<br>1h (R/W) = Interrupt was generated |
| 6 | PCO | R | 0h | Position counter overflow interrupt flag<br>Reset type: SYSRSn<br>0h (R/W) = No interrupt generated<br>1h (R/W) = Interrupt was generated |
| 5 | PCU | R | 0h | Position counter underflow interrupt flag<br>Reset type: SYSRSn<br>0h (R/W) = No interrupt generated<br>1h (R/W) = Interrupt was generated |
| 4 | WTO | R | 0h | Watchdog timeout interrupt flag<br>Reset type: SYSRSn<br>0h (R/W) = No interrupt generated<br>1h (R/W) = Interrupt was generated |
| 3 | QDC | R | 0h | Quadrature direction change interrupt flag<br>Reset type: SYSRSn<br>0h (R/W) = No interrupt generated<br>1h (R/W) = Interrupt was generated |



Copyright © 2024 Texas Instruments Incorporated

**Table 17-22. QFLG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 2 | PHE | R | 0h | Quadrature phase error interrupt flag<br>Reset type: SYSRSn<br>0h (R/W) = No interrupt generated<br>1h (R/W) = Interrupt was generated |
| 1 | PCE | R | 0h | Position counter error interrupt flag<br>Reset type: SYSRSn<br>0h (R/W) = No interrupt generated<br>1h (R/W) = Interrupt was generated |
| 0 | INT | R | 0h | Global interrupt status flag<br>Reset type: SYSRSn<br>0h (R/W) = No interrupt generated<br>1h (R/W) = Interrupt was generated |



Copyright © 2024 Texas Instruments Incorporated

### 17.10.2.18 QCLR Register (Offset = 1Ah) [Reset = 0000h]

QCLR is shown in Figure 17-38 and described in Table 17-23.

Return to the Summary Table.

QEP Interrupt Clear

**Figure 17-38. QCLR Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | UTO | IEL | SEL | PCM |
| R-0h | | | | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PCR | PCO | PCU | WTO | QDC | PHE | PCE | INT |
| R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h | R-0/W1S-0h |

**Table 17-23. QCLR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | RESERVED | R | 0h | Reserved |
| 11 | UTO | R-0/W1S | 0h | Clear unit time out interrupt flag<br>Reset type: SYSRSn<br>0h (R/W) = No effect<br>1h (R/W) = Clears the interrupt flag |
| 10 | IEL | R-0/W1S | 0h | Clear index event latch interrupt flag<br>Reset type: SYSRSn<br>0h (R/W) = No effect<br>1h (R/W) = Clears the interrupt flag |
| 9 | SEL | R-0/W1S | 0h | Clear strobe event latch interrupt flag<br>Reset type: SYSRSn<br>0h (R/W) = No effect<br>1h (R/W) = Clears the interrupt flag |
| 8 | PCM | R-0/W1S | 0h | Clear eQEP compare match event interrupt flag<br>Reset type: SYSRSn<br>0h (R/W) = No effect<br>1h (R/W) = Clears the interrupt flag |
| 7 | PCR | R-0/W1S | 0h | Clear position-compare ready interrupt flag<br>Reset type: SYSRSn<br>0h (R/W) = No effect<br>1h (R/W) = Clears the interrupt flag |
| 6 | PCO | R-0/W1S | 0h | Clear position counter overflow interrupt flag<br>Reset type: SYSRSn<br>0h (R/W) = No effect<br>1h (R/W) = Clears the interrupt flag |
| 5 | PCU | R-0/W1S | 0h | Clear position counter underflow interrupt flag<br>Reset type: SYSRSn<br>0h (R/W) = No effect<br>1h (R/W) = Clears the interrupt flag |
| 4 | WTO | R-0/W1S | 0h | Clear watchdog timeout interrupt flag<br>Reset type: SYSRSn<br>0h (R/W) = No effect<br>1h (R/W) = Clears the interrupt flag |
| 3 | QDC | R-0/W1S | 0h | Clear quadrature direction change interrupt flag<br>Reset type: SYSRSn<br>0h (R/W) = No effect<br>1h (R/W) = Clears the interrupt flag |



Copyright © 2024 Texas Instruments Incorporated

**Table 17-23. QCLR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 2 | PHE | R-0/W1S | 0h | Clear quadrature phase error interrupt flag<br>Reset type: SYSRSn<br>0h (R/W) = No effect<br>1h (R/W) = Clears the interrupt flag |
| 1 | PCE | R-0/W1S | 0h | Clear position counter error interrupt flag<br>Reset type: SYSRSn<br>0h (R/W) = No effect<br>1h (R/W) = Clears the interrupt flag |
| 0 | INT | R-0/W1S | 0h | Global interrupt clear flag<br>Reset type: SYSRSn<br>0h (R/W) = No effect<br>1h (R/W) = Clears the interrupt flag |



Copyright © 2024 Texas Instruments Incorporated

### 17.10.2.19 QFRC Register (Offset = 1Bh) [Reset = 0000h]

QFRC is shown in Figure 17-39 and described in Table 17-24.

Return to the Summary Table.

QEP Interrupt Force

**Figure 17-39. QFRC Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | UTO | IEL | SEL | PCM |
| R-0h | | | | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PCR | PCO | PCU | WTO | QDC | PHE | PCE | RESERVED |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R-0h |

**Table 17-24. QFRC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | RESERVED | R | 0h | Reserved |
| 11 | UTO | R/W | 0h | Force unit time out interrupt<br>Reset type: SYSRSn<br>0h (R/W) = No effect<br>1h (R/W) = Force the interrupt |
| 10 | IEL | R/W | 0h | Force index event latch interrupt<br>Reset type: SYSRSn<br>0h (R/W) = No effect<br>1h (R/W) = Force the interrupt |
| 9 | SEL | R/W | 0h | Force strobe event latch interrupt<br>Reset type: SYSRSn<br>0h (R/W) = No effect<br>1h (R/W) = Force the interrupt |
| 8 | PCM | R/W | 0h | Force position-compare match interrupt<br>Reset type: SYSRSn<br>0h (R/W) = No effect<br>1h (R/W) = Force the interrupt |
| 7 | PCR | R/W | 0h | Force position-compare ready interrupt<br>Reset type: SYSRSn<br>0h (R/W) = No effect<br>1h (R/W) = Force the interrupt |
| 6 | PCO | R/W | 0h | Force position counter overflow interrupt<br>Reset type: SYSRSn<br>0h (R/W) = No effect<br>1h (R/W) = Force the interrupt |
| 5 | PCU | R/W | 0h | Force position counter underflow interrupt<br>Reset type: SYSRSn<br>0h (R/W) = No effect<br>1h (R/W) = Force the interrupt |
| 4 | WTO | R/W | 0h | Force watchdog time out interrupt<br>Reset type: SYSRSn<br>0h (R/W) = No effect<br>1h (R/W) = Force the interrupt |
| 3 | QDC | R/W | 0h | Force quadrature direction change interrupt<br>Reset type: SYSRSn<br>0h (R/W) = No effect<br>1h (R/W) = Force the interrupt |



Copyright © 2024 Texas Instruments Incorporated

**Table 17-24. QFRC Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 2 | PHE | R/W | 0h | Force quadrature phase error interrupt<br>Reset type: SYSRSn<br>0h (R/W) = No effect<br>1h (R/W) = Force the interrupt |
| 1 | PCE | R/W | 0h | Force position counter error interrupt<br>Reset type: SYSRSn<br>0h (R/W) = No effect<br>1h (R/W) = Force the interrupt |
| 0 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 17.10.2.20 QEPSTS Register (Offset = 1Ch) [Reset = 0000h]

QEPSTS is shown in Figure 17-40 and described in Table 17-25.

Return to the Summary Table.

QEP Status

**Figure 17-40. QEPSTS Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| UPEVNT | FIDF | QDF | QDLF | COEF | CDEF | FIMF | PCEF |
| R/W-0h | R-0h | R-0h | R-0h | R/W-0h | R/W-0h | R/W-0h | R-0h |

**Table 17-25. QEPSTS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R | 0h | Reserved |
| 7 | UPEVNT | R/W | 0h | Unit position event flag<br>Reset type: SYSRSn<br>0h (R/W) = No unit position event detected<br>1h (R/W) = Unit position event detected. Write 1 to clear |
| 6 | FIDF | R | 0h | Direction on the first index marker<br>Status of the direction is latched on the first index event marker.<br>Reset type: SYSRSn<br>0h (R/W) = Counter-clockwise rotation (or reverse movement) on the first index event<br>1h (R/W) = Clockwise rotation (or forward movement) on the first index event |
| 5 | QDF | R | 0h | Quadrature direction flag<br>Reset type: SYSRSn<br>0h (R/W) = Counter-clockwise rotation (or reverse movement)<br>1h (R/W) = Clockwise rotation (or forward movement) |
| 4 | QDLF | R | 0h | eQEP direction latch flag<br>Reset type: SYSRSn<br>0h (R/W) = Counter-clockwise rotation (or reverse movement) on index event marker<br>1h (R/W) = Clockwise rotation (or forward movement) on index event marker |
| 3 | COEF | R/W | 0h | Capture overflow error flag<br>Reset type: SYSRSn<br>0h (R/W) = Overflow has not occurred.<br>1h (R/W) = Overflow occurred in eQEP Capture timer (QEPCTMR). This bit is cleared by writing a '1'. |
| 2 | CDEF | R/W | 0h | Capture direction error flag<br>Reset type: SYSRSn<br>0h (R/W) = Capture direction error has not occurred.<br>1h (R/W) = Direction change occurred between the capture position event. This bit is cleared by writing a '1'. |
| 1 | FIMF | R/W | 0h | First index marker flag<br>Note: Once this flag has been set, if the flag is cleared the flag will not be set again until the module is reset by a peripheral or system reset.<br>Reset type: SYSRSn<br>0h (R/W) = First index pulse has not occurred.<br>1h (R/W) = Set by first occurrence of index pulse. This bit is cleared by writing a '1'. |



Copyright © 2024 Texas Instruments Incorporated

**Table 17-25. QEPSTS Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | PCEF | R | 0h | Position counter error flag.<br>This bit is not sticky and it is updated for every index event.<br>Reset type: SYSRSn<br>0h (R/W) = No error occurred during the last index transition<br>1h (R/W) = Position counter error |



Copyright © 2024 Texas Instruments Incorporated

### 17.10.2.21 QCTMR Register (Offset = 1Dh) [Reset = 0000h]

QCTMR is shown in Figure 17-41 and described in Table 17-26.

Return to the Summary Table.

QEP Capture Timer

**Figure 17-41. 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 17-26. QCTMR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | QCTMR | R/W | 0h | This register provides time base for edge capture unit.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 17.10.2.22 QCPRD Register (Offset = 1Eh) [Reset = 0000h]

QCPRD is shown in Figure 17-42 and described in Table 17-27.

Return to the Summary Table.

QEP Capture Period

**Figure 17-42. 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 17-27. QCPRD Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | QCPRD | R/W | 0h | This register holds the period count value between the last successive eQEP position events<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 17.10.2.23 QCTMRLAT Register (Offset = 1Fh) [Reset = 0000h]

QCTMRLAT is shown in Figure 17-43 and described in Table 17-28.

Return to the Summary Table.

QEP Capture Latch

**Figure 17-43. 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 17-28. QCTMRLAT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | QCTMRLAT | R | 0h | The eQEP capture timer value can be latched into this register on two events viz., unit timeout event, reading the eQEP position counter.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 17.10.2.24 QCPRDLAT Register (Offset = 20h) [Reset = 0000h]

QCPRDLAT is shown in Figure 17-44 and described in Table 17-29.

Return to the Summary Table.

QEP Capture Period Latch

**Figure 17-44. 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 17-29. QCPRDLAT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | QCPRDLAT | R | 0h | eQEP capture period value can be latched into this register on two events viz., unit timeout event, reading the eQEP position counter.<br>Reset type: SYSRSn |

### *17.10.3 EQEP Registers to Driverlib Functions*

**Table 17-30. EQEP Registers to Driverlib Functions**

| File | Driverlib Function |
|---|---|
| **QPOSCNT** | |
| eqep.h | EQEP_getPosition |
| eqep.h | EQEP_setPosition |
| **QPOSINIT** | |
| eqep.h | EQEP_setInitialPosition |
| **QPOSMAX** | |
| eqep.h | EQEP_setPositionCounterConfig |
| **QPOSCMP** | |
| eqep.c | EQEP_setCompareConfig |
| **QPOSILAT** | |
| eqep.h | EQEP_getIndexPositionLatch |
| **QPOSSLAT** | |
| eqep.h | EQEP_getStrobePositionLatch |
| **QPOSLAT** | |
| eqep.h | EQEP_getPositionLatch |
| **QUTMR** | |
| - | |
| **QUPRD** | |
| eqep.h | EQEP_loadUnitTimer |
| eqep.h | EQEP_enableUnitTimer |
| **QWDTMR** | |
| eqep.h | EQEP_setWatchdogTimerValue |
| eqep.h | EQEP_getWatchdogTimerValue |
| **QWDPRD** | |
| eqep.h | EQEP_enableWatchdog |



Copyright © 2024 Texas Instruments Incorporated

**Table 17-30. EQEP Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| **QDECCTL** | |
| eqep.c | EQEP_setCompareConfig |
| eqep.c | EQEP_setInputPolarity |
| eqep.h | EQEP_setDecoderConfig |
| **QEPCTL** | |
| 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 |
| **QCAPCTL** | |
| eqep.h | EQEP_setCaptureConfig |
| eqep.h | EQEP_enableCapture |
| eqep.h | EQEP_disableCapture |
| **QPOSCTL** | |
| eqep.c | EQEP_setCompareConfig |
| eqep.h | EQEP_enableCompare |
| eqep.h | EQEP_disableCompare |
| eqep.h | EQEP_setComparePulseWidth |
| **QEINT** | |
| eqep.h | EQEP_enableInterrupt |
| eqep.h | EQEP_disableInterrupt |
| **QFLG** | |
| eqep.h | EQEP_getInterruptStatus |
| eqep.h | EQEP_getError |
| **QCLR** | |
| eqep.h | EQEP_clearInterruptStatus |
| **QFRC** | |
| eqep.h | EQEP_forceInterrupt |
| **QEPSTS** | |
| eqep.h | EQEP_getDirection |
| eqep.h | EQEP_getStatus |
| eqep.h | EQEP_clearStatus |
| **QCTMR** | |
| eqep.h | EQEP_getCaptureTimer |
| eqep.h | EQEP_getCaptureTimerLatch |
| **QCPRD** | |
| eqep.h | EQEP_getCapturePeriod |
| eqep.h | EQEP_getCapturePeriodLatch |



Copyright © 2024 Texas Instruments Incorporated

**Table 17-30. EQEP Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| **QCTMRLAT** | |
| eqep.h | EQEP_getCaptureTimerLatch |
| **QCPRDLAT** | |
| eqep.h | EQEP_getCapturePeriodLatch |



Copyright © 2024 Texas Instruments Incorporated

*Chapter 18*

# *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.


**18.1 Introduction** ..... 2251
**18.2 System-Level Integration** ..... 2253
**18.3 SPI Operation** ..... 2257
**18.4 Programming Procedure** ..... 2268
**18.5 Software** ..... 2274
**18.6 SPI Registers** ..... 2277





Copyright © 2024 Texas Instruments Incorporated

## 18.1 Introduction

### *18.1.1 Features*

The SPI module features include:

- SPISOMI: SPI slave-output/master-input pin
- SPISIMO: SPI slave-input/master-output pin
- $\overline{\text{SPISTE}}$: SPI slave transmit-enable pin
- SPICLK: SPI serial-clock pin

**Note**

All four pins can be used as GPIO if the SPI module is not used.

- Two operational modes: Master and Slave
- Baud rate: 125 different programmable rates. The maximum baud rate that can be employed is limited by the maximum speed of the I/O buffers used on the SPI pins. See the device 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
- DMA support
- High-speed mode
- Delayed transmit control
- 3-wire SPI mode
- $\overline{\text{SPISTE}}$ inversion for digital audio interface receive mode on devices with two SPI modules

### *18.1.2 SPI Related Collateral*

**Foundational Materials**

- C2000 Academy - SPI
- *KeyStone Architecture Serial Peripheral Interface (SPI)*

**Getting Started Materials**

- SPI: Microcontroller overview (Video)



Copyright © 2024 Texas Instruments Incorporated

### 18.1.3 Block Diagram

Figure 18-1 shows the SPI CPU interfaces.

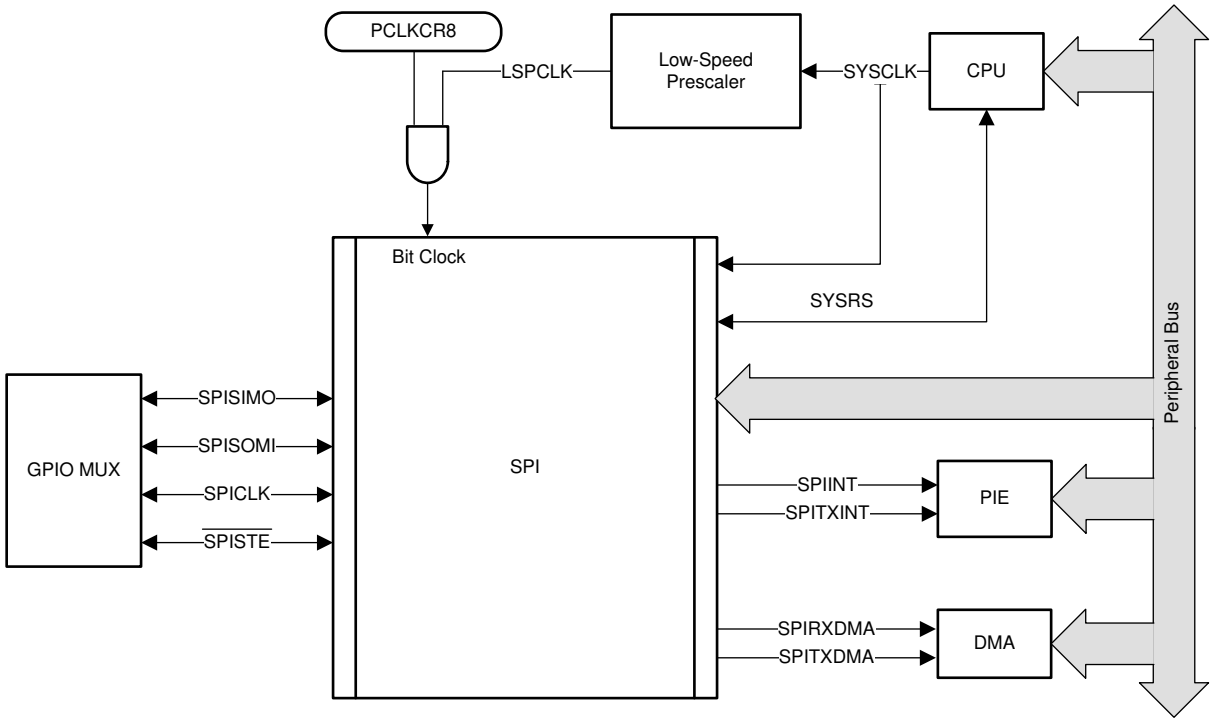



**Figure 18-1. SPI CPU Interface**



Copyright © 2024 Texas Instruments Incorporated

## 18.2 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.

### *18.2.1 SPI Module Signals*

Table 18-1 classifies and provides a summary of the SPI module signals.

**Table 18-1. SPI Module Signal Summary**

| Signal Name | Description |
|---|---|
| **External Signals** | |
| SPICLK | SPI clock |
| SPISIMO | SPI slave in, master out |
| SPISOMI | SPI slave out, master in |
| $\overline{\text{SPISTE}}$ | SPI slave transmit enable |
| **Control** | |
| SPI Clock Rate | LSPCLK |
| **Interrupt Signals** | |
| 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 |
| **DMA Triggers** | |
| SPITXDMA | Transmit request to DMA |
| SPIRXDMA | Receive request to DMA |

#### Special Considerations

The $\overline{\text{SPISTE}}$ signal provides the ability to gate any spurious clock and data pulses when the SPI is in slave mode. 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 18.6.

#### 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.



Copyright © 2024 Texas Instruments Incorporated

### 18.2.2 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.

#### 18.2.2.1 GPIOs Required for High-Speed Mode

The high-speed mode of the SPI is available on the specified GPIO mux options in the device data sheet. 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.

### 18.2.3 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 18-2 and Table 18-2 show how these control bits influence the SPI interrupt generation.



Copyright © 2024 Texas Instruments Incorporated

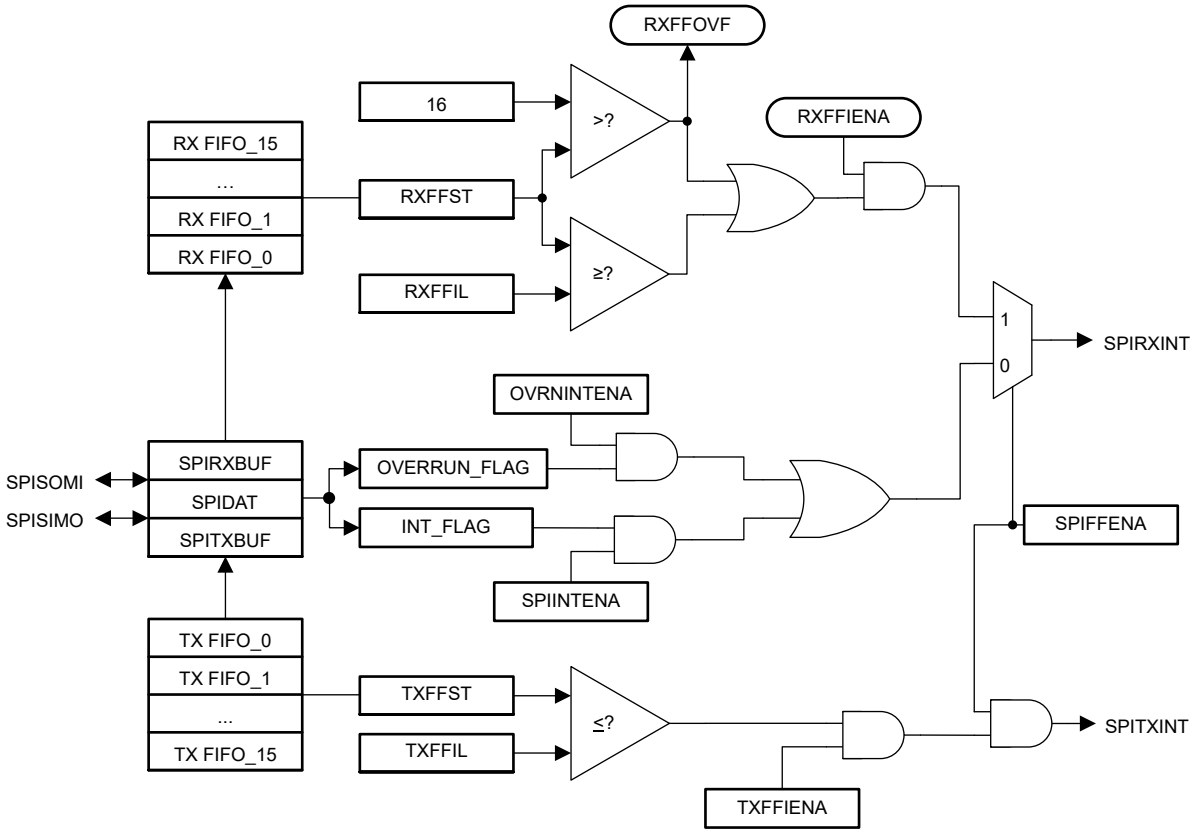



**Figure 18-2. SPI Interrupt Flags and Enable Logic Generation**

**Table 18-2. SPI Interrupt Flag Modes**

| FIFO Options | SPI Interrupt Source | Interrupt Flags | Interrupt Enables | FIFO Enable (SPIFFENA) | Interrupt Line (1) |
|---|---|---|---|---|---|
| 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.



Copyright © 2024 Texas Instruments Incorporated

### 18.2.4 DMA Support

Both the CPU and DMA have access to the SPI data registers using the internal peripheral bus. This access is limited to 16-bit register reads and writes. Each SPI module can generate two DMA events, SPITXDMA and SPIRXDMA. The DMA events are controlled by configuring the SPIFFTX.TXFFIL and SPIFFRX.RXFFIL appropriately. SPITXDMA activates when TXFFST is less than the interrupt level (TXFFIL). SPIRXDMA activates when RXFFST is greater than or equal to the interrupt level (RXFFIL).

The SPI must have FIFO enhancements enabled for the DMA triggers to be generated.

For more information on configuring the SPI for DMA transfers, refer to Section 18.3.8.

Figure 18-3 is a block diagram showing the DMA trigger generation from the SPI module.

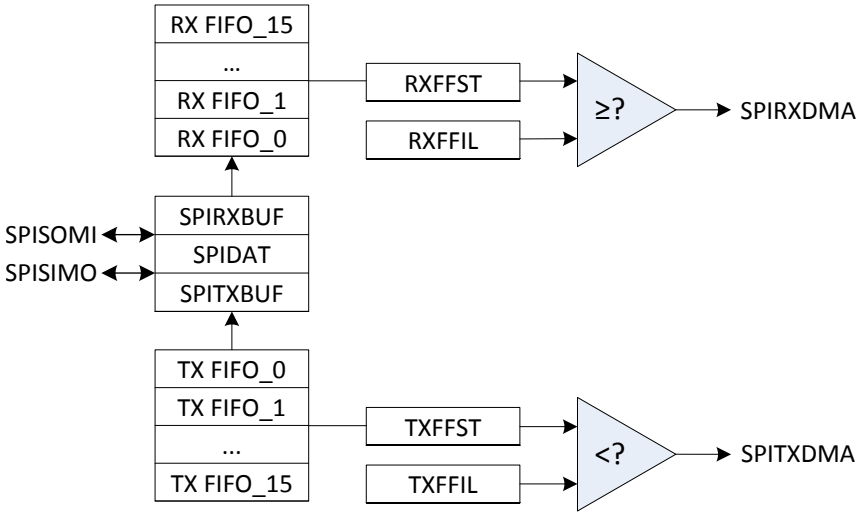



**Figure 18-3. SPI DMA Trigger Diagram**



Copyright © 2024 Texas Instruments Incorporated

## 18.3 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.

### *18.3.1 Introduction to Operation*

Figure 18-4 shows typical connections of the SPI for communications between two controllers: a master and a slave.

The master transfers data by sending the SPICLK signal. For both the slave and the master, data is shifted out of the shift registers on one edge of the SPICLK and latched into the shift register on the opposite SPICLK clock edge. If the CLK_PHASE bit is high, data is transmitted and received a half-cycle before the SPICLK transition. As a result, both controllers send and receive data simultaneously. The application software determines whether the data is meaningful or dummy data. There are three possible methods for data transmission:

- Master sends data; slave sends dummy data.
- Master sends data; slave sends data.
- Master sends dummy data; slave sends data.

The master can initiate data transfer at any time because 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.

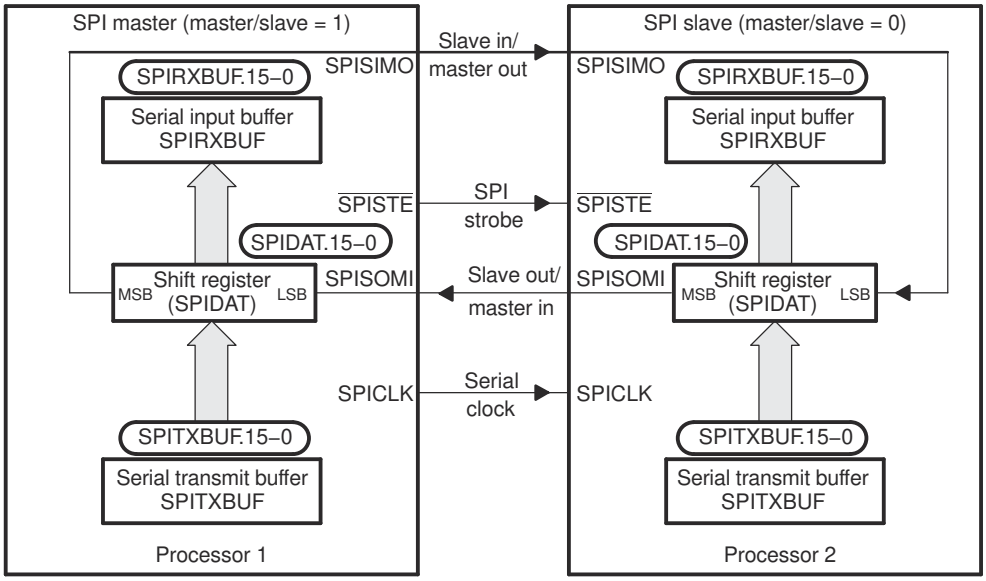



**Figure 18-4. SPI Master/Slave Connection**



Copyright © 2024 Texas Instruments Incorporated

### 18.3.2 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 18-5 is a block diagram of the SPI in master mode. The block diagram shows the basic control blocks available in SPI master mode.



Copyright © 2024 Texas Instruments Incorporated

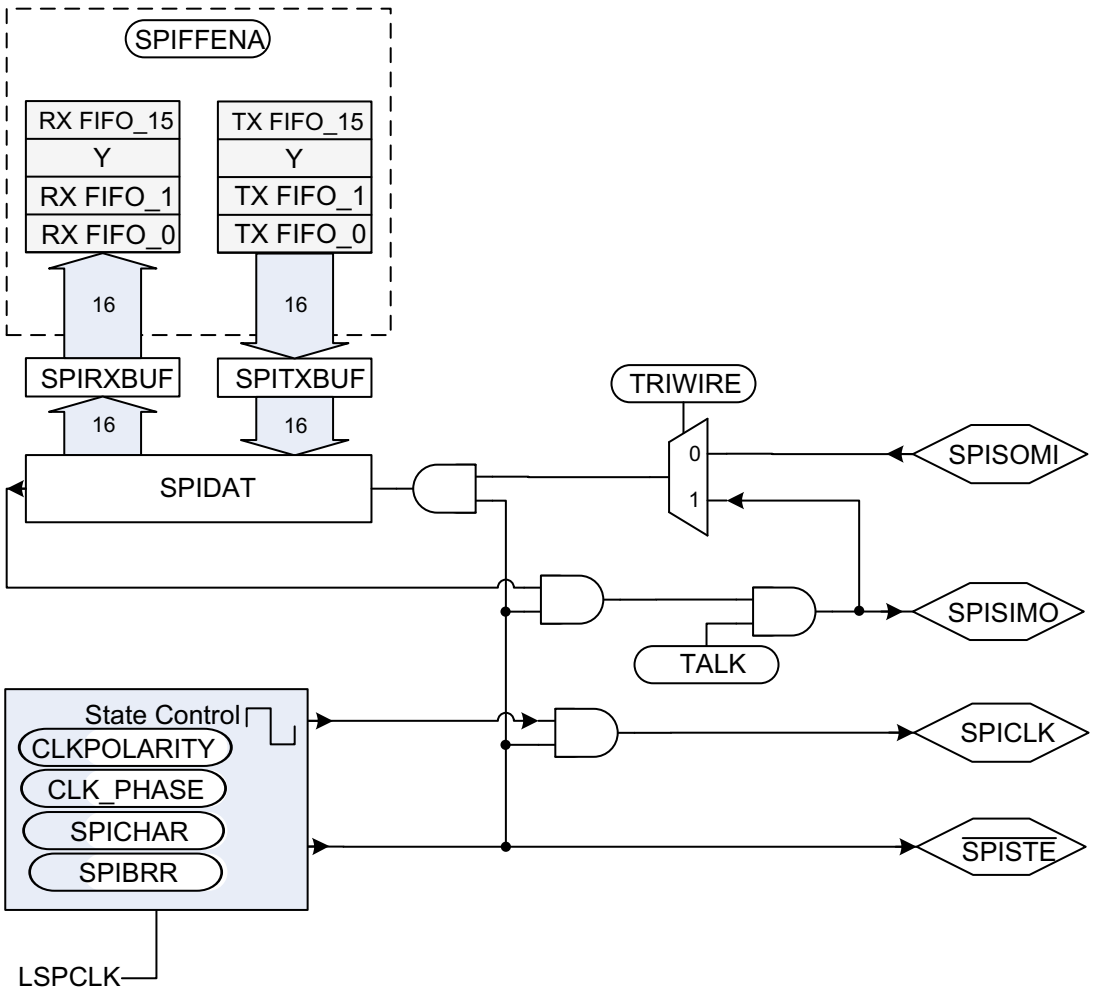



**Figure 18-5. SPI Module Master Configuration**

### *18.3.3 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



Copyright © 2024 Texas Instruments Incorporated

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 18-6 is a block diagram of the SPI in slave mode. The block diagram shows the basic control blocks available in SPI slave mode.

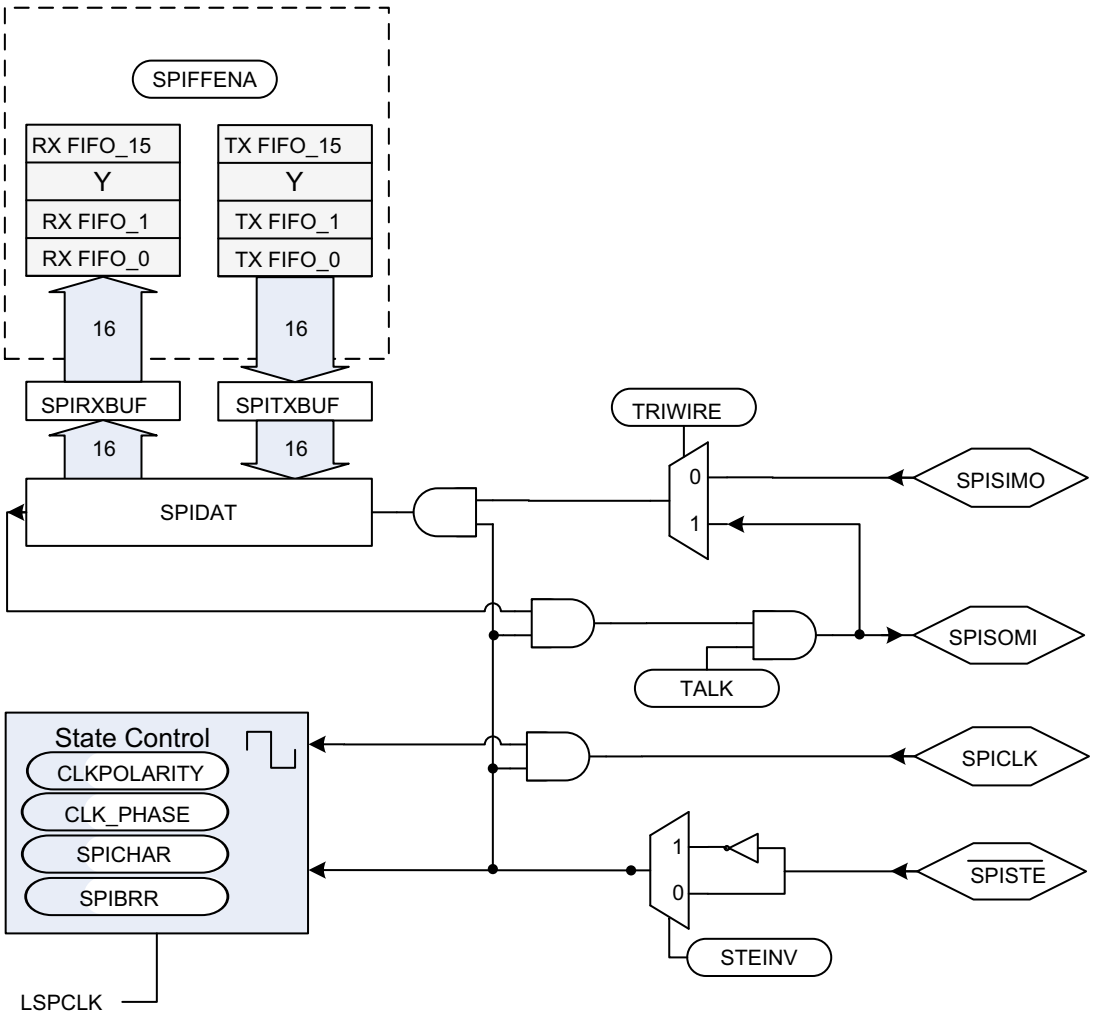



**Figure 18-6. SPI Module Slave Configuration**



Copyright © 2024 Texas Instruments Incorporated

### 18.3.4 Data Format

The four-bit SPICHAR 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 18-1).

***Example 18-1. Transmission of Bit from SPIRXBUF***

Conditions:

1. Transmission character length = 1 bit (specified in SPICHAR 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(1) ← (RXed) |
| **SPIRXBUF (after transmission)** | | | | | | | | | | | | | | | | |
| | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | x(1) |

(1) x = 1, if SPISOMI data is high; x = 0, if SPISOMI data is low; master mode is assumed.



Copyright © 2024 Texas Instruments Incorporated

### 18.3.5 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 18-2 shows how to determine the SPI baud rates.

Example 18-3 shows how to calculate the baud rate of the SPI module in standard SPI mode (HS_MODE = 0).

***Example 18-2. 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 18-3. Baud Rate Calculation in Non-High Speed Mode (HS_MODE = 0)***

$$SPI\ Baud\ Rate = \frac{LSPCLK}{SPIBRR + 1'} \quad LSPCLK = 50\ MHz$$

$$= \frac{50 \times 10^6}{3 + 1}$$

$$= 12.5\ Mbps$$



Copyright © 2024 Texas Instruments Incorporated

### 18.3.6 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:

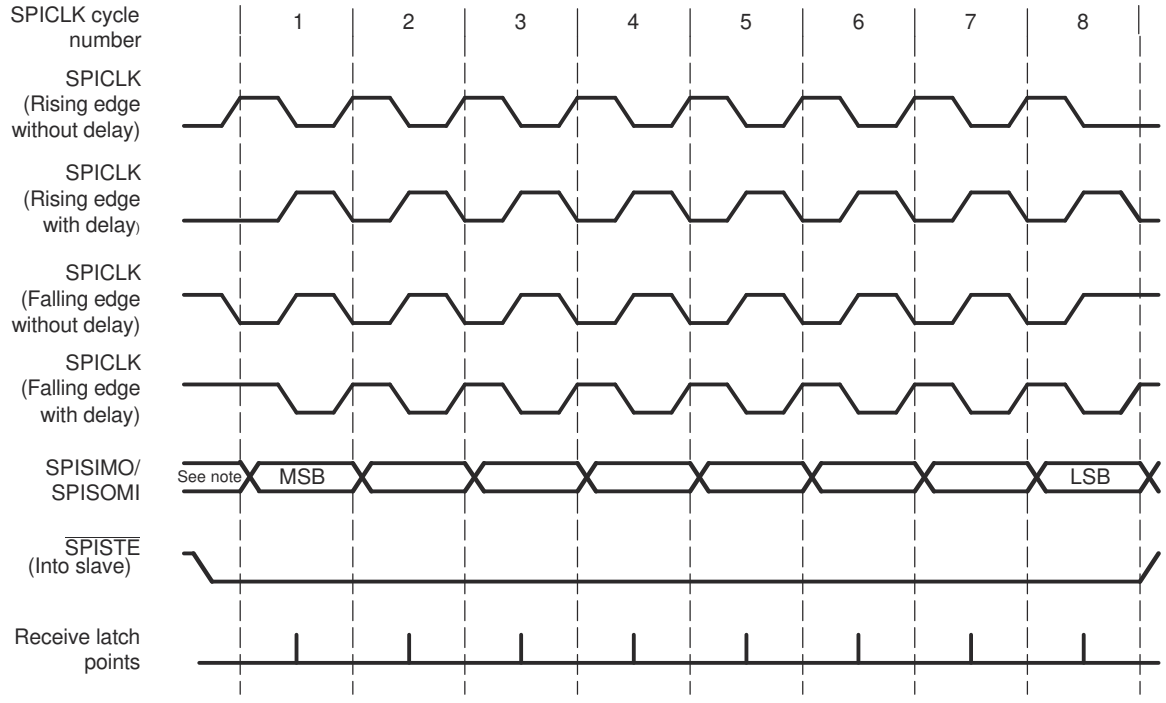
- Falling Edge Without Delay. The SPI transmits data on the falling edge of the SPICLK and receives data on the rising edge of the SPICLK.
- Falling Edge With Delay. The SPI transmits data one half-cycle ahead of the falling edge of the SPICLK signal and receives data on the falling edge of the SPICLK 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 18-3. Examples of these four clocking schemes relative to transmitted and received data are shown in Figure 18-7.

**Table 18-3. SPI Clocking Scheme Selection Guide**

| SPICLK Scheme | CLKPOLARITY(1) | CLK_PHASE(1) |
|---|---|---|
| Rising edge without delay | 0 | 0 |
| Rising edge with delay | 0 | 1 |
| Falling edge without delay | 1 | 0 |
| Falling edge with delay | 1 | 1 |

(1) The description of 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 18-7. SPICLK Signal Options**



Copyright © 2024 Texas Instruments Incorporated

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 18-8.

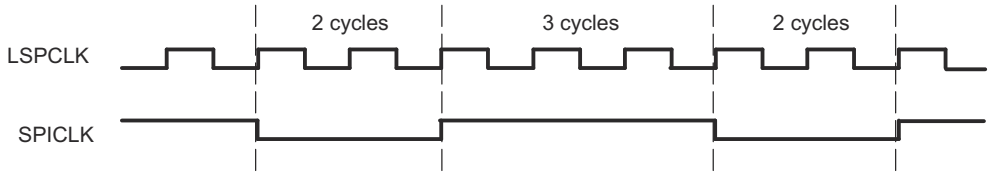



**Figure 18-8. SPI: SPICLK-LSPCLK Characteristic when (BRR + 1) is Odd, BRR > 3, and CLKPOLARITY = 1**

### *18.3.7 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 18.2.3.
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 and DMA triggers. The transmit interrupt (SPITXINT) is generated whenever the transmit FIFO status bits (TXFFST) match (less than or equal to) the interrupt trigger level bits (TXFFIL). The receive interrupt (SPIRXINT) is generated whenever the receive FIFO status bits (RXFFST) match (greater than or equal to) the interrupt trigger level RXFFIL. This provides a programmable interrupt trigger for transmit and receive sections of the SPI. The default value for these trigger level bits is 0x11111 for receive FIFO and 0x00000 for transmit FIFO, respectively.



Copyright © 2024 Texas Instruments Incorporated

### 18.3.8 SPI DMA Transfers

#### 18.3.8.1 Transmitting Data Using SPI with DMA

When using the DMA with the TX FIFO, the DMA Burst Size (DMA_BURST_SIZE) must be no greater than 16 – TXFFIL, to prevent the DMA from writing to an already full FIFO. This leads to data loss and is not recommended.

For complete data transmission, follow these steps:

1. Calculate the total number of words to be transmitted. [NUM_WORDS]
2. Decide the transmit FIFO level. [TXFFIL]
3. Calculate the total number of DMA transfers. [DMA_TRANSFER_SIZE]
4. Calculate the size of the DMA Burst. [DMA_BURST_SIZE]
5. Configure DMA using calculated values.
6. Configure SPI with FIFO using the calculated values.

To transfer 128 words to SPI using the DMA:

NUM_WORDS: 128

TXFFIL: 8

DMA_TRANSFER_SIZE: (NUM_WORDS /TXFFIL) – 1 = (128/8) – 1 = 15 (16 transfers)

DMA_BURST_SIZE: (16 – TXFFIL) – 1 = (16 – 8) – 1 = 7 (8 words per burst)

**Note**

Avoid setting TXFFIL to 0h or 10h to make sure of proper DMA configuration.

#### 18.3.8.2 Receiving Data Using SPI with DMA

When using the DMA with the RX FIFO, the DMA Burst Size (BURST_SIZE) must be no greater than RXFFIL to prevent the DMA from reading from an empty FIFO. To make sure that the DMA correctly receives all data from the RX FIFO, the DMA Burst Size can equal RXFFIL and also be an integer divisor of the total number of SPI transmissions.

For complete data reception, follow these steps:

1. Calculate the total number of words to be received. [NUM_WORDS]
2. Calculate the necessary FIFO level [RXFFIL]
3. Calculate the total number of DMA transfers. [DMA_TRANSFER_SIZE]
4. Calculate the size of the DMA Burst. [DMA_BURST_SIZE]
5. Configure DMA using the calculated values.
6. Configure SPI with FIFO using the calculated values.

To receive 200 words from SPI using the DMA:

NUM_WORDS = 200

RXFFIL: 4

DMA_TRANSFER_SIZE: (NUM_WORDS /RXFFIL) – 1 = (200/4) – 1 = 49 (50 transfers)

DMA_BURST_SIZE = RXFFIL-1 = 3 (4 words per burst)

**Note**

Avoid setting RXFFIL to 0h to make sure proper DMA configuration.



Copyright © 2024 Texas Instruments Incorporated

### 18.3.9 SPI High-Speed Mode

The SPI module is capable of reaching full-duplex communication speeds up to LSPCLK/4 (where LSPCLK equals SYSCLK). For the maximum rated speed, refer 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 18.2.2.1 for more information.

### 18.3.10 SPI 3-Wire Mode Description

SPI 3-wire mode allows for SPI communication over three pins instead of the normal four pins.

In master mode, if the TRIWIRE bit is set, enabling 3-wire SPI mode, SPISIMOx becomes the bi-directional SPIMOMIx (SPI master out, master in) pin, and SPISOMIx is no longer used by the SPI. In slave mode, if the TRIWIRE bit is set, SPISOMIx becomes the bi-directional SPISISOx (SPI slave in, slave out) pin, and SPISIMOx is no longer used by the SPI.

Table 18-4 indicates the pin function differences between 3-wire and 4-wire SPI mode for a master and slave SPI.

**Table 18-4. 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 |
| SPISIMOx | SPIMOMIx | Free |
| SPISOMIx | Free | SPISISOx |

Because in 3-wire mode, the receive and transmit paths within the SPI are connected, any data transmitted by the SPI module is also received. 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 18-9 and Figure 18-10 illustrate 3-wire master and slave mode.

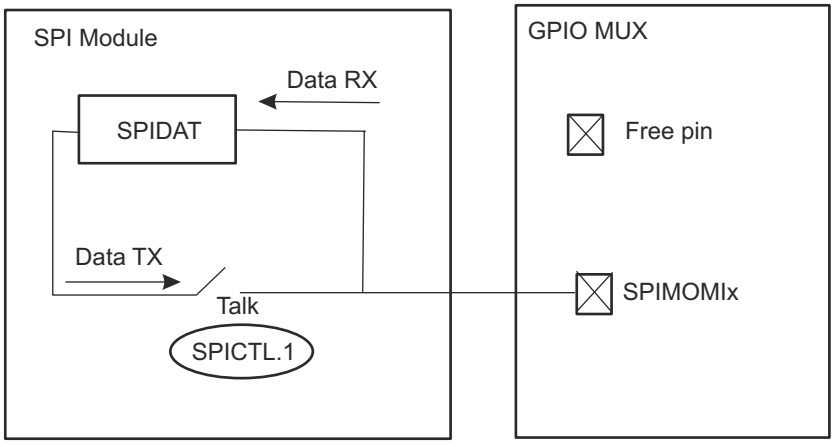



**Figure 18-9. SPI 3-wire Master Mode**



Copyright © 2024 Texas Instruments Incorporated

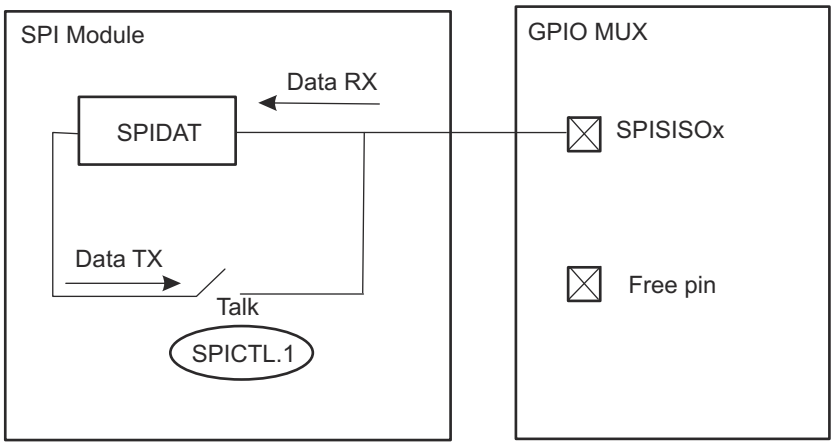



**Figure 18-10. SPI 3-wire Slave Mode**

Table 18-5 indicates how data is received or transmitted in the various SPI modes while the TALK bit is set or cleared.

**Table 18-5. 3-Wire SPI Pin Configuration**

| Pin Mode | SPIPRI[TRIWIRE] | SPICTL[TALK] | SPISIMO | SPISOMI |
|---|---|---|---|---|
| | | **Master Mode** | | |
| 4-wire | 0 | X | TX | RX |
| 3-pin mode | 1 | 0 | RX | Disconnect from SPI |
| | | 1 | TX/RX | |
| | | **Slave Mode** | | |
| 4-wire | 0 | X | RX | TX |
| 3-pin mode | 1 | 0 | Disconnect from SPI | RX |
| | | 1 | | TX/RX |



Copyright © 2024 Texas Instruments Incorporated

## 18.4 Programming Procedure

This section describes the procedure for configuring the SPI for the various modes of operation.

### *18.4.1 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

### *18.4.2 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 (SPICHAR).
   - 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.



Copyright © 2024 Texas Instruments Incorporated

### 18.4.3 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) = 25MHz

```
SpiaRegs.SPIBRR = 0x3;
```

There are no other differences in the configuration from normal SPI operation. Sending and receiving data, DMA operation, and interrupts operates without change.



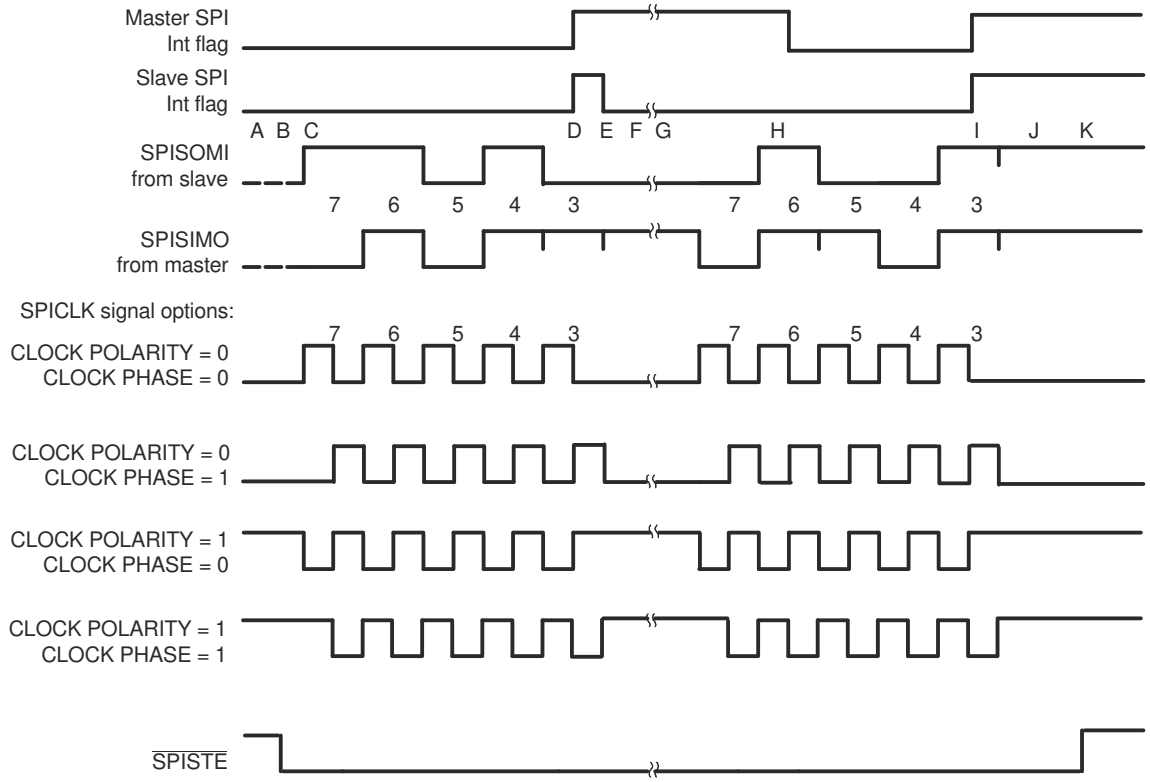
Copyright © 2024 Texas Instruments Incorporated

### 18.4.4 Data Transfer Example

The timing diagram shown in Figure 18-11 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 18-8) shares similar characterizations with Figure 18-11 except that the data transfer is one LSPCLK cycle longer per bit during the low pulse (CLKPOLARITY = 0) or during the high pulse (CLKPOLARITY = 1) of the SPICLK.

Figure 18-11 is applicable for 8-bit SPI only and is not for C28x devices that are capable of working with 16-bit data. The figure is shown for illustrative purposes only.




A. Slave writes 0D0h to SPIDAT and waits for the master to shift out the data.
B. Master sets the slave $\overline{\text{SPISTE}}$ signal low (active).
C. Master writes 058h to SPIDAT, which starts the transmission procedure.
D. First byte is finished and sets the interrupt flags.
E. Slave reads 0Bh from the SPIRXBUF (right-justified).
F. Slave writes 04Ch to SPIDAT and waits for the master to shift out the data.
G. Master writes 06Ch to SPIDAT, which starts the transmission procedure.
H. Master reads 01Ah from the SPIRXBUF (right-justified).
I. Second byte is finished and sets the interrupt flags.
J. Master reads 89h and the slave reads 8Dh from the respective SPIRXBUF. After the user software masks off the unused bits, the master receives 09h and the slave receives 0Dh.
K. Master clears the slave $\overline{\text{SPISTE}}$ signal high (inactive).

**Figure 18-11. Five Bits per Character**



Copyright © 2024 Texas Instruments Incorporated

### 18.4.5 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, $\overline{\text{SPISTE}}$x, 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 18-4. 3-Wire Master Mode Transmit***

```
Uint16 data;
Uint16 dummy;
     SpiaRegs.SPICTL.bit.TALK = 1;              // Enable Transmit path
     SpiaRegs.SPITXBUF = data; // Master transmits data
     while(SpiaRegs.SPISTS.bit.INT_FLAG !=1) {} // Waits until data rx'd
     dummy = SpiaRegs.SPIRXBUF;                 // Clears junk data 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 18-5. 3-Wire Master Mode Receive***

```
Uint16 rdata;
Uint16 dummy;
     SpiaRegs.SPICTL.bit.TALK = 0;              // Disable Transmit path
     SpiaRegs.SPITXBUF = dummy;                 // Send dummy to start tx
     // NOTE: because TALK = 0, data does not tx onto SPISIMOA pin
     while(SpiaRegs.SPISTS.bit.INT_FLAG !=1) {} // Wait until data received
     rdata = SpiaRegs.SPIRXBUF;                 // Master reads data
```

In 3-wire slave mode, SPICLKx, SPISTEx, and SPISOMIx pins must be configured as SPI pins (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 18-6. 3-Wire Slave Mode Transmit***

```
Uint16 data;
Uint16 dummy;
     SpiaRegs.SPICTL.bit.TALK = 1;              // Enable Transmit path
     SpiaRegs.SPITXBUF = data;                  // Slave transmits data
     while(SpiaRegs.SPISTS.bit.INT_FLAG !=1) {} // Wait until data rx'd
     dummy = SpiaRegs.SPIRXBUF;                 // Clears junk data
```



Copyright © 2024 Texas Instruments Incorporated

As in 3-wire master mode, the TALK bit must be cleared to 0. Otherwise, the slave receives data normally.

***Example 18-7. 3-Wire Slave Mode Receive***

```
Uint16 rdata;
    SpiaRegs.SPICTL.bit.TALK = 0;              // Disable Transmit path
    while(SpiaRegs.SPISTS.bit.INT_FLAG !=1) {} // Waits until data rx'd
    rdata = SpiaRegs.SPIRXBUF;                 // Slave reads data
```



Copyright © 2024 Texas Instruments Incorporated

### 18.4.6 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 18-12.

**Note**

This configuration is only applicable to slave mode (MASTER_SLAVE = 0). When the SPI is configured as master (MASTER_SLAVE = 1), the STEINV bit has no effect on the $\overline{\text{SPISTE}}$ pin.

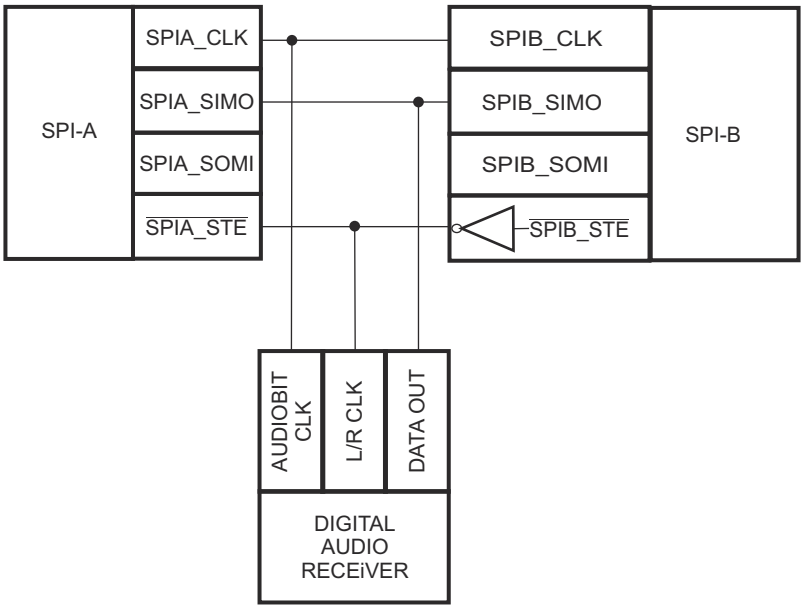



**Figure 18-12. SPI Digital Audio Receiver Configuration Using Two SPIs**

Standard C28x SPI timing requirements limit the number of digital audio interface formats supported using the 2-SPI configuration with the STEINV bit. See 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 18-13.

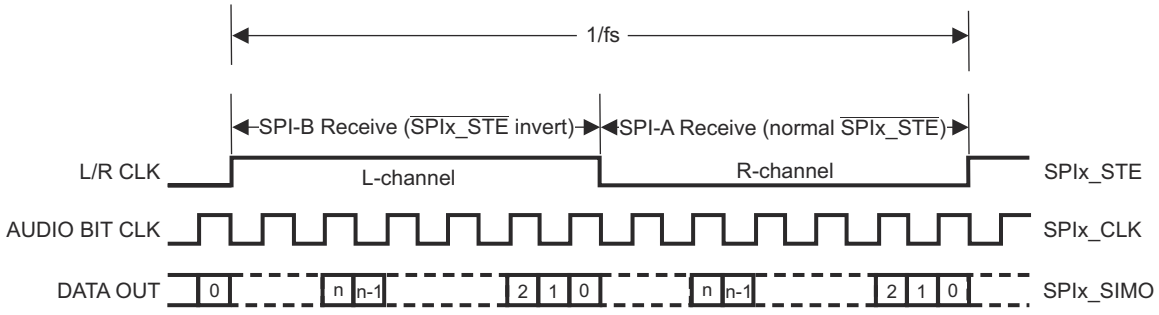



**Figure 18-13. Standard Right-Justified Digital Audio Data Format**



Copyright © 2024 Texas Instruments Incorporated

## 18.5 Software

### *18.5.1 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 C2000Ware Examples.

#### 18.5.1.1 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 configure 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.

This example project has support for migration across our C2000 device families. If you are wanting to build this project from launchpad or controlCARD, please specify in the .syscfg file the board you're using. At any time you can select another device to migrate this example. *External Connections*

- None

*Watch Variables*

- *sData* - Data to send
- *rData* - Received data

#### 18.5.1.2 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.

This example project has support for migration across our C2000 device families. If you are wanting to build this project from launchpad or controlCARD, please specify in the .syscfg file the board you're using. At any time you can select another device to migrate this example. *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

#### 18.5.1.3 SPI Digital External Loopback without FIFO Interrupts

FILE: spi_ex3_external_loopback.c



Copyright © 2024 Texas Instruments Incorporated

This program uses the external loopback between two SPI modules. Both the SPI FIFOs and interrupts are not used in this example. SPIA is configured as a peripheral and SPI B is configured as controller. This example demonstrates full duplex communication where both controller and peripheral transmits and receives data simultaneously.

This example project has support for migration across our C2000 device families. If you are wanting to build this project from launchpad or controlCARD, please specify in the .syscfg file the board you're using. At any time you can select another device to migrate this example. *External Connections*
Refer to SysConfig for external connections (GPIO pin numbers) specific to each device

*Watch Variables*
- *TxData_SPIA* - Data send from SPIA (peripheral)
- *TxData_SPIB* - Data send from SPIB (controller)
- *RxData_SPIA* - Data received by SPIA (peripheral)
- *RxData_SPIB* - Data received by SPIB (controller)

#### 18.5.1.4 SPI Digital External Loopback with FIFO Interrupts

FILE: spi_ex4_external_loopback_fifo_interrupts.c

This program uses the external loopback between two SPI modules. Both the SPI FIFOs and their interrupts are used. SPIA is configured as a peripheral and receives data from SPI B which is configured as a controller.

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.

This example project has support for migration across our C2000 device families. If you are wanting to build this project from launchpad or controlCARD, please specify in the .syscfg file the board you're using. At any time you can select another device to migrate this example. *External Connections*
Refer to SysConfig for external connections (GPIO pin numbers) specific to each device

*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

#### 18.5.1.5 SPI Digital Loopback with DMA

FILE: spi_ex5_loopback_dma.c

This program uses the internal loopback test mode of the SPI module. Both DMA interrupts and the SPI FIFOs are used. When the SPI transmit FIFO has enough space (as indicated by its FIFO level interrupt signal), the DMA will transfer data from global variable sData into the FIFO. This will be transmitted to the receive FIFO via the internal loopback.

When enough data has been placed in the receive FIFO (as indicated by its FIFO level interrupt signal), the DMA will transfer the data from the FIFO into global variable rData.

When all data has been placed into rData, a check of the validity of the data will be performed in one of the DMA channels' ISRs.



Copyright © 2024 Texas Instruments Incorporated

This example project has support for migration across our C2000 device families. If you are wanting to build this project from launchpad or controlCARD, please specify in the .syscfg file the board you're using. At any time you can select another device to migrate this example. *External Connections*

- None

*Watch Variables*

- *sData* - Data to send
- *rData* - Received data

#### 18.5.1.6 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*
*ExternalConnections*

- Connect external SPI EEPROM
- Connect GPIO16 (PICO) to external EEPROM SI pin
- Connect GPIO17 (POCI) to external EEPROM SO pin
- Connect GPIO18 (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

#### 18.5.1.7 SPI DMA EEPROM

FILE: spi_ex7_eeprom_dma.c

This program will write 8 bytes to EEPROM and read them back. The device communicates with the EEPROM via SPI using DMA and specific opcodes. This example is written to work with the SPI Serial EEPROM AT25128/256. Note: Runtime updation of character length when CS pin is not controlled by the SPI module can lead to unpredictable behaviour

*External Connections*
*ExternalConnections*

- Connect external SPI EEPROM
- Connect GPIO16 (PICO) to external EEPROM SI pin
- Connect GPIO17 (POCI) to external EEPROM SO pin
- Connect GPIO18 (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
- SPI_DMA_Handle.RXdata - Data that is read back from EEPROM when number of received bytes is less than 4
- SPI_DMA_Handle.pSPIRXDMA->pbuffer - Start address of received data from EEPROM
- error - Error count



Copyright © 2024 Texas Instruments Incorporated

## 18.6 SPI Registers

This section describes the Serial Peripheral Interface registers. It is important to note that the SPI registers only allow 16-bit accesses.

### *18.6.1 SPI Base Addresses*

**Table 18-6. SPI Base Address Table**

| Device Registers | Register Name | Start Address | End Address |
|---|---|---|---|
| SpiaRegs | SPI_REGS | 0x0000_6100 | 0x0000_610F |
| SpibRegs | SPI_REGS | 0x0000_6110 | 0x0000_611F |
| SpicRegs | SPI_REGS | 0x0000_6120 | 0x0000_612F |



Copyright © 2024 Texas Instruments Incorporated

### 18.6.2 SPI_REGS Registers

Table 18-7 lists the memory-mapped registers for the SPI_REGS registers. All register offset addresses not listed in Table 18-7 should be considered as reserved locations and the register contents should not be modified.

**Table 18-7. SPI_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | SPICCR | SPI Configuration Control Register | | Go |
| 1h | SPICTL | SPI Operation Control Register | | Go |
| 2h | SPISTS | SPI Status Register | | Go |
| 4h | SPIBRR | SPI Baud Rate Register | | Go |
| 6h | SPIRXEMU | SPI Emulation Buffer Register | | Go |
| 7h | SPIRXBUF | SPI Serial Input Buffer Register | | Go |
| 8h | SPITXBUF | SPI Serial Output Buffer Register | | Go |
| 9h | SPIDAT | SPI Serial Data Register | | Go |
| Ah | SPIFFTX | SPI FIFO Transmit Register | | Go |
| Bh | SPIFFRX | SPI FIFO Receive Register | | Go |
| Ch | SPIFFCT | SPI FIFO Control Register | | Go |
| Fh | SPIPRI | SPI Priority Control Register | | Go |

Complex bit access types are encoded to fit into small table cells. Table 18-8 shows the codes that are used for access types in this section.

**Table 18-8. SPI_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| RC | R<br>C | Read<br>to Clear |
| Write Type | | |
| W | W | Write |
| W1C | W<br>1C | Write<br>1 to clear |
| Reset or Default Value | | |
| *-n* | | Value after reset or the default value |



Copyright © 2024 Texas Instruments Incorporated

### 18.6.2.1 SPICCR Register (Offset = 0h) [Reset = 0000h]

SPICCR is shown in Figure 18-14 and described in Table 18-9.

Return to the Summary Table.

SPICCR controls the setup of the SPI for operation.

**Figure 18-14. SPICCR Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SPISWRESET | CLKPOLARITY | HS_MODE | SPILBK | SPICHAR | | | |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | | | |

**Table 18-9. SPICCR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R | 0h | Reserved |
| 7 | SPISWRESET | R/W | 0h | SPI Software Reset<br>When changing configuration, you should clear this bit before the changes and set this bit before resuming operation.<br>Reset type: SYSRSn<br>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.<br>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. |
| 6 | CLKPOLARITY | R/W | 0h | Shift Clock Polarity<br>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.<br>Reset type: SYSRSn<br>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:<br>- 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.<br>- 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.<br>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:<br>- 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.<br>- 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. |



Copyright © 2024 Texas Instruments Incorporated

**Table 18-9. SPICCR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 5 | HS_MODE | R/W | 0h | High Speed Mode Enable Bits<br>This bit determines if the High Speed mode is enabled. The correct GPIOs should be selected in the GPxGMUX/GPxMUX registers.<br>Reset type: SYSRSn<br>0h (R/W) = SPI High Speed mode disabled. This is the default value after reset.<br>1h (R/W) = SPI High Speed mode enabled, |
| 4 | SPILBK | R/W | 0h | SPI Loopback Mode Select<br>Loopback mode allows module validation during device testing. This mode is valid only in master mode of the SPI.<br>Reset type: SYSRSn<br>0h (R/W) = SPI loopback mode disabled. This is the default value after reset.<br>1h (R/W) = SPI loopback mode enabled, SIMO/SOMI lines are connected internally. Used for module self-tests. |
| 3-0 | SPICHAR | R/W | 0h | Character Length Control Bits<br>These four bits determine the number of bits to be shifted in or SPI CHAR0 out as a single character during one shift sequence.<br>SPICHAR = Word length - 1<br>Reset type: SYSRSn<br>0h (R/W) = 1-bit word<br>1h (R/W) = 2-bit word<br>7h (R/W) = 8-bit word<br>Fh (R/W) = 16-bit word |



Copyright © 2024 Texas Instruments Incorporated

### 18.6.2.2 SPICTL Register (Offset = 1h) [Reset = 0000h]

SPICTL is shown in Figure 18-15 and described in Table 18-10.

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 18-15. SPICTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | OVERRUNINTENA | CLK_PHASE | MASTER_SLAVE | TALK | SPIINTENA |
| R-0h | | | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 18-10. SPICTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-5 | RESERVED | R | 0h | Reserved |
| 4 | OVERRUNINTENA | R/W | 0h | Overrun Interrupt Enable<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = Disable RECEIVER OVERRUN interrupts.<br>1h (R/W) = Enable RECEIVER_OVERRUN interrupts. |
| 3 | CLK_PHASE | R/W | 0h | SPI Clock Phase Select<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = Normal SPI clocking scheme, depending on the CLOCK POLARITY bit (SPICCR.6).<br>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<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = SPI is configured as a slave.<br>1h (R/W) = SPI is configured as a master. |



Copyright © 2024 Texas Instruments Incorporated

**Table 18-10. SPICTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1 | TALK | R/W | 0h | Transmit Enable<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = Disables transmission:<br>- 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.<br>- 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.<br>1h (R/W) = Enables transmission For the 4-pin option, ensure to enable the receiver's SPISTEn input pin. |
| 0 | SPIINTENA | R/W | 0h | SPI Interrupt Enable<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = Disables the interrupt.<br>1h (R/W) = Enables the interrupt. |



Copyright © 2024 Texas Instruments Incorporated

### 18.6.2.3 SPISTS Register (Offset = 2h) [Reset = 0000h]

SPISTS is shown in Figure 18-16 and described in Table 18-11.

Return to the Summary Table.

SPISTS contains interrupt and status bits.

**Figure 18-16. SPISTS Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OVERRUN_FLAG | INT_FLAG | BUFFULL_FLAG | RESERVED | | | | |
| W1C-0h | RC-0h | R-0h | R-0h | | | | |

**Table 18-11. SPISTS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R | 0h | Reserved |
| 7 | OVERRUN_FLAG | W1C | 0h | SPI Receiver Overrun Flag<br>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:<br>- Writing a 1 to this bit<br>- Writing a 0 to SPI SW RESET (SPICCR.7)<br>- Resetting the system<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = A receive overrun condition has not occurred.<br>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).<br>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. |



Copyright © 2024 Texas Instruments Incorporated

**Table 18-11. SPISTS Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 6 | INT_FLAG | RC | 0h | SPI Interrupt Flag<br>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:<br>- Reading SPIRXBUF<br>- Writing a 0 to SPI SW RESET (SPICCR.7)<br>- Resetting the system<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = No full words have been received or transmitted.<br>1h (R/W) = Indicates that the SPI has completed sending or receiving the last bit and is ready to be serviced. |
| 5 | BUFFULL_FLAG | R | 0h | SPI Transmit Buffer Full Flag<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = Transmit buffer is not full.<br>1h (R/W) = Transmit buffer is full. |
| 4-0 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 18.6.2.4 SPIBRR Register (Offset = 4h) [Reset = 0000h]

SPIBRR is shown in Figure 18-17 and described in Table 18-12.

Return to the Summary Table.

SPIBRR contains the bits used for baud-rate selection.

**Figure 18-17. SPIBRR Register**

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

**Table 18-12. SPIBRR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-7 | RESERVED | R | 0h | Reserved |
| 6-0 | SPI_BIT_RATE | R/W | 0h | SPI Baud Rate Control<br>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.)<br>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.<br>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:<br>For SPIBRR = 3 to 127: SPI Baud Rate = LSPCLK / (SPIBRR + 1)<br>For SPIBRR = 0, 1, or 2: SPI Baud Rate = LSPCLK / 4<br>Reset type: SYSRSn<br>3h (R/W) = SPI Baud Rate = LSPCLK/4<br>4h (R/W) = SPI Baud Rate = LSPCLK/5<br>7Eh (R/W) = SPI Baud Rate = LSPCLK/127<br>7Fh (R/W) = SPI Baud Rate = LSPCLK/128 |



Copyright © 2024 Texas Instruments Incorporated

### 18.6.2.5 SPIRXEMU Register (Offset = 6h) [Reset = 0000h]

SPIRXEMU is shown in Figure 18-18 and described in Table 18-13.

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 18-18. SPIRXEMU Register**

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

**Table 18-13. SPIRXEMU Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | ERXBn | R | 0h | Emulation Buffer Received Data<br>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.<br>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.<br>It is recommended that you view SPIRXEMU in the normal emulator run mode.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 18.6.2.6 SPIRXBUF Register (Offset = 7h) [Reset = 0000h]

SPIRXBUF is shown in Figure 18-19 and described in Table 18-14.

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 18-19. SPIRXBUF Register**

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

**Table 18-14. SPIRXBUF Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RXBn | R | 0h | Received Data<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 18.6.2.7 SPITXBUF Register (Offset = 8h) [Reset = 0000h]

SPITXBUF is shown in Figure 18-20 and described in Table 18-15.

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 18-20. SPITXBUF Register**

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

**Table 18-15. SPITXBUF Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | TXBn | R/W | 0h | Transmit Data Buffer<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 18.6.2.8 SPIDAT Register (Offset = 9h) [Reset = 0000h]

SPIDAT is shown in Figure 18-21 and described in Table 18-16.

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 18-21. SPIDAT Register**

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

**Table 18-16. SPIDAT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | SDATn | R/W | 0h | Serial Data Shift Register<br>- It provides data to be output on the serial output pin if the TALK bit (SPICTL.1) is set.<br>- 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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 18.6.2.9 SPIFFTX Register (Offset = Ah) [Reset = A000h]

SPIFFTX is shown in Figure 18-22 and described in Table 18-17.

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 18-22. SPIFFTX Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| SPIRST | SPIFFENA | TXFIFO | TXFFST | | | | |
| R/W-1h | R/W-0h | R/W-1h | R-0h | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TXFFINT | TXFFINTCLR | TXFFIENA | TXFFIL | | | | |
| R-0h | W-0h | R/W-0h | R/W-0h | | | | |

**Table 18-17. SPIFFTX Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | SPIRST | R/W | 1h | SPI Reset<br>Reset type: SYSRSn<br>0h (R/W) = Write 0 to reset the SPI transmit and receive channels. The SPI FIFO register configuration bits will be left as is.<br>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<br>Reset type: SYSRSn<br>0h (R/W) = SPI FIFO enhancements are disabled.<br>1h (R/W) = SPI FIFO enhancements are enabled. |
| 13 | TXFIFO | R/W | 1h | TX FIFO Reset<br>Reset type: SYSRSn<br>0h (R/W) = Write 0 to reset the FIFO pointer to zero, and hold in reset.<br>1h (R/W) = Release transmit FIFO from reset. |
| 12-8 | TXFFST | R | 0h | Transmit FIFO Status<br>Reset type: SYSRSn<br>0h (R/W) = Transmit FIFO is empty.<br>1h (R/W) = Transmit FIFO has 1 word.<br>2h (R/W) = Transmit FIFO has 2 words.<br>10h (R/W) = Transmit FIFO has 16 words, which is the maximum.<br>1Fh (R/W) = Reserved. |
| 7 | TXFFINT | R | 0h | TX FIFO Interrupt Flag<br>Reset type: SYSRSn<br>0h (R/W) = TXFIFO interrupt has not occurred, This is a read-only bit.<br>1h (R/W) = TXFIFO interrupt has occurred, This is a read-only bit. |
| 6 | TXFFINTCLR | W | 0h | TXFIFO Interrupt Clear<br>Reset type: SYSRSn<br>0h (R/W) = Write 0 has no effect on TXFIFINT flag bit, Bit reads back a zero.<br>1h (R/W) = Write 1 to clear SPIFFTX[TXFFINT] flag. |
| 5 | TXFFIENA | R/W | 0h | TX FIFO Interrupt Enable<br>Reset type: SYSRSn<br>0h (R/W) = TX FIFO interrupt based on TXFFIL match (less than or equal to) will be disabled.<br>1h (R/W) = TX FIFO interrupt based on TXFFIL match (less than or equal to) will be enabled. |



Copyright © 2024 Texas Instruments Incorporated

**Table 18-17. SPIFFTX Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 4-0 | TXFFIL | R/W | 0h | Transmit FIFO Interrupt Level Bits<br>Transmit FIFO will generate interrupt when the FIFO status bits (TXFFST4-0) and FIFO level bits (TXFFIL4-0 ) match (less than or equal to).<br>Reset type: SYSRSn<br>0h (R/W) = A TX FIFO interrupt request is generated when there are no words remaining in the TX buffer.<br>1h (R/W) = A TX FIFO interrupt request is generated when there is 1 word or no words remaining in the TX buffer.<br>2h (R/W) = A TX FIFO interrupt request is generated when there is 2 words or fewer remaining in the TX buffer.<br>10h (R/W) = A TX FIFO interrupt request is generated when there are 16 words or fewer remaining in the TX buffer.<br>1Fh (R/W) = Reserved. |



Copyright © 2024 Texas Instruments Incorporated

### 18.6.2.10 SPIFFRX Register (Offset = Bh) [Reset = 201Fh]

SPIFFRX is shown in Figure 18-23 and described in Table 18-18.

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 18-23. SPIFFRX Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RXFFOVF | RXFFOVFCLR | RXFIFORESET | RXFFST | | | | |
| R-0h | W-0h | R/W-1h | R-0h | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RXFFINT | RXFFINTCLR | RXFFIENA | RXFFIL | | | | |
| R-0h | W-0h | R/W-0h | R/W-1Fh | | | | |

**Table 18-18. SPIFFRX Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RXFFOVF | R | 0h | Receive FIFO Overflow Flag<br>Reset type: SYSRSn<br>0h (R/W) = Receive FIFO has not overflowed. This is a read-only bit.<br>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<br>Reset type: SYSRSn<br>0h (R/W) = Write 0 does not affect RXFFOVF flag bit, Bit reads back a zero.<br>1h (R/W) = Write 1 to clear SPIFFRX[RXFFOVF]. |
| 13 | RXFIFORESET | R/W | 1h | Receive FIFO Reset<br>Reset type: SYSRSn<br>0h (R/W) = Write 0 to reset the FIFO pointer to zero, and hold in reset.<br>1h (R/W) = Re-enable receive FIFO operation. |
| 12-8 | RXFFST | R | 0h | Receive FIFO Status<br>Reset type: SYSRSn<br>0h (R/W) = Receive FIFO is empty.<br>1h (R/W) = Receive FIFO has 1 word.<br>2h (R/W) = Receive FIFO has 2 words.<br>10h (R/W) = Receive FIFO has 16 words, which is the maximum.<br>1Fh (R/W) = Reserved. |
| 7 | RXFFINT | R | 0h | Receive FIFO Interrupt Flag<br>Reset type: SYSRSn<br>0h (R/W) = RXFIFO interrupt has not occurred. This is a read-only bit.<br>1h (R/W) = RXFIFO interrupt has occurred. This is a read-only bit. |
| 6 | RXFFINTCLR | W | 0h | Receive FIFO Interrupt Clear<br>Reset type: SYSRSn<br>0h (R/W) = Write 0 has no effect on RXFIFINT flag bit, Bit reads back a zero.<br>1h (R/W) = Write 1 to clear SPIFFRX[RXFFINT] flag |
| 5 | RXFFIENA | R/W | 0h | RX FIFO Interrupt Enable<br>Reset type: SYSRSn<br>0h (R/W) = RX FIFO interrupt based on RXFFIL match (greater than or equal to) will be disabled.<br>1h (R/W) = RX FIFO interrupt based on RXFFIL match (greater than or equal to) will be enabled. |



Copyright © 2024 Texas Instruments Incorporated

**Table 18-18. SPIFFRX Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 4-0 | RXFFIL | R/W | 1Fh | Receive FIFO Interrupt Level Bits<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = A RX FIFO interrupt request is generated when there is 0 or more words in the RX buffer.<br>1h (R/W) = A RX FIFO interrupt request is generated when there are 1 or more words in the RX buffer.<br>2h (R/W) = A RX FIFO interrupt request is generated when there are 2 or more words in the RX buffer.<br>10h (R/W) = A RX FIFO interrupt request is generated when there are 16 words in the RX buffer.<br>1Fh (R/W) = Reserved. |



Copyright © 2024 Texas Instruments Incorporated

### 18.6.2.11 SPIFFCT Register (Offset = Ch) [Reset = 0000h]

SPIFFCT is shown in Figure 18-24 and described in Table 18-19.

Return to the Summary Table.

SPIFFCT controls the FIFO transmit delay bits.

**Figure 18-24. SPIFFCT Register**

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

**Table 18-19. SPIFFCT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R | 0h | Reserved |
| 7-0 | TXDLY | R/W | 0h | FIFO Transmit Delay Bits<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = The next word in the TX FIFO buffer is transferred to SPITXBUF immediately upon completion of transmission of the previous word.<br>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.<br>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.<br>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. |



Copyright © 2024 Texas Instruments Incorporated

### 18.6.2.12 SPIPRI Register (Offset = Fh) [Reset = 0000h]

SPIPRI is shown in Figure 18-25 and described in Table 18-20.

Return to the Summary Table.

SPIPRI controls auxiliary functions for the SPI including emulation control, SPISTE inversion, and 3-wire control.

**Figure 18-25. SPIPRI Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | RESERVED | SOFT | FREE | RESERVED | | STEINV | TRIWIRE |
| R-0h | R/W-0h | R/W-0h | R/W-0h | R-0h | | R/W-0h | R/W-0h |

**Table 18-20. 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 | Emulation Soft Run<br>This bit only has an effect when the FREE bit is 0.<br>Reset type: SYSRSn<br>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.<br>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.<br>Standard SPI mode: Stop after transmitting the words in the shift register and buffer. That is, after TXBUF and SPIDAT are empty.<br>In FIFO mode: Stop after transmitting the words in the shift register and buffer. That is, after TX FIFO and SPIDAT are empty. |
| 4 | FREE | R/W | 0h | Emulation Free Run<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = Emulation mode is selected by the SOFT bit<br>1h (R/W) = Free run, continue SPI operation regardless of suspend or when the suspend occurred. |
| 3-2 | RESERVED | R | 0h | Reserved |
| 1 | STEINV | R/W | 0h | SPISTEn Inversion Bit<br>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.<br>This bit is only applicable to slave mode. Writing to this bit while configured as master (MASTER_SLAVE = 1) has no effect<br>Reset type: SYSRSn<br>0h (R/W) = SPISTEn is active low (normal)<br>1h (R/W) = SPISTE is active high (inverted) |



Copyright © 2024 Texas Instruments Incorporated

**Table 18-20. SPIPRI Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | TRIWIRE | R/W | 0h | SPI 3-wire Mode Enable<br>Reset type: SYSRSn<br>0h (R/W) = Normal 4-wire SPI mode.<br>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. |

### *18.6.3 SPI Registers to Driverlib Functions*

**Table 18-21. SPI Registers to Driverlib Functions**

| File | Driverlib Function |
|---|---|
| **SPICCR** | |
| 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 |
| **SPICTL** | |
| spi.c | SPI_setConfig |
| spi.c | SPI_enableInterrupt |
| spi.c | SPI_disableInterrupt |
| spi.h | SPI_enableTalk |
| spi.h | SPI_disableTalk |
| **SPISTS** | |
| spi.c | SPI_getInterruptStatus |
| spi.c | SPI_clearInterruptStatus |
| spi.h | SPI_writeDataBlockingNonFIFO |
| spi.h | SPI_readDataBlockingNonFIFO |
| **SPIBRR** | |
| spi.c | SPI_setConfig |
| spi.c | SPI_setBaudRate |
| **SPIRXEMU** | |
| spi.h | SPI_readRxEmulationBuffer |
| **SPIRXBUF** | |
| spi.h | SPI_readDataNonBlocking |
| spi.h | SPI_readDataBlockingFIFO |
| spi.h | SPI_readDataBlockingNonFIFO |
| **SPITXBUF** | |
| spi.h | SPI_writeDataNonBlocking |
| spi.h | SPI_writeDataBlockingFIFO |



Copyright © 2024 Texas Instruments Incorporated

**Table 18-21. SPI Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| spi.h | SPI_writeDataBlockingNonFIFO |
| **SPIDAT** | |
| - | |
| **SPIFFTX** | |
| 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 |
| **SPIFFRX** | |
| 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 |
| **SPIFFCT** | |
| spi.h | SPI_setTxFifoTransmitDelay |
| **SPIPRI** | |
| spi.h | SPI_enableTriWire |
| spi.h | SPI_disableTriWire |
| spi.h | SPI_setPTESignalPolarity |
| spi.h | SPI_setEmulationMode |



Copyright © 2024 Texas Instruments Incorporated

Chapter 19

# 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.


**19.1 Introduction** ..... 2299
**19.2 Architecture** ..... 2300
**19.3 SCI Module Signal Summary** ..... 2300
**19.4 Configuring Device Pins** ..... 2302
**19.5 Multiprocessor and Asynchronous Communication Modes** ..... 2302
**19.6 SCI Programmable Data Format** ..... 2303
**19.7 SCI Multiprocessor Communication** ..... 2304
**19.8 Idle-Line Multiprocessor Mode** ..... 2305
**19.9 Address-Bit Multiprocessor Mode** ..... 2307
**19.10 SCI Communication Format** ..... 2308
**19.11 SCI Port Interrupts** ..... 2311
**19.12 SCI Baud Rate Calculations** ..... 2312
**19.13 SCI Enhanced Features** ..... 2313
**19.14 Software** ..... 2316
**19.15 SCI Registers** ..... 2316





Copyright © 2024 Texas Instruments Incorporated

## 19.1 Introduction

The SCI interfaces are shown in Figure 19-1.

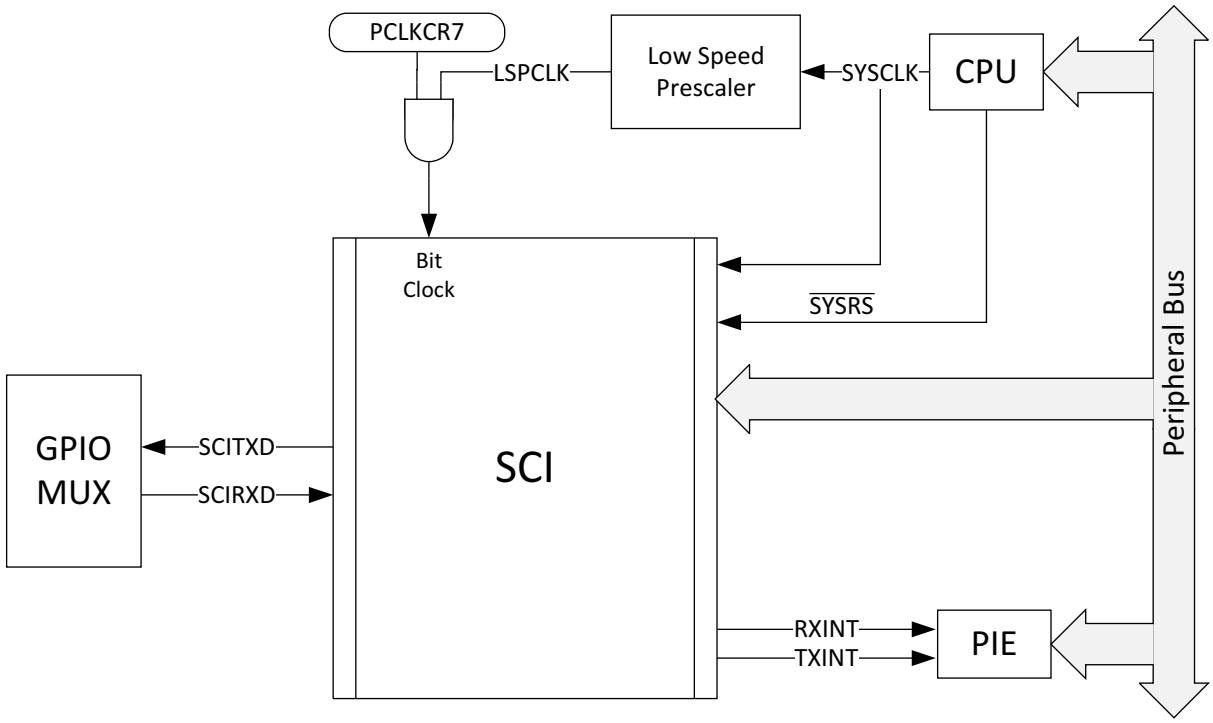



**Figure 19-1. SCI CPU Interface**

### *19.1.1 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



Copyright © 2024 Texas Instruments Incorporated

### 19.1.2 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?

### 19.1.3 Block Diagram

Figure 19-2 shows the SCI module block diagram. The SCI port operation is configured and controlled by the registers listed in Section 19.15.

## 19.2 Architecture

The major elements used in full-duplex operation are shown in Figure 19-2 and include:

- A transmitter (TX) and the major registers (upper half of Figure 19-2):
  - 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 19-2):
  - 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.

## 19.3 SCI Module Signal Summary

A summarized description of each SCI signal name is shown in Table 19-1.

**Table 19-1. SCI Module Signal Summary**

| Signal Name | Description |
|---|---|
| **External signals** | |
| SCIRXD | SCI Asynchronous Serial Port receive data |
| SCITXD | SCI Asynchronous Serial Port transmit data |
| **Control** | |
| Baud clock | LSPCLK Prescaled clock |
| **Interrupt signals** | |
| TXINT | Transmit interrupt |
| RXINT | Receive interrupt |



Copyright © 2024 Texas Instruments Incorporated

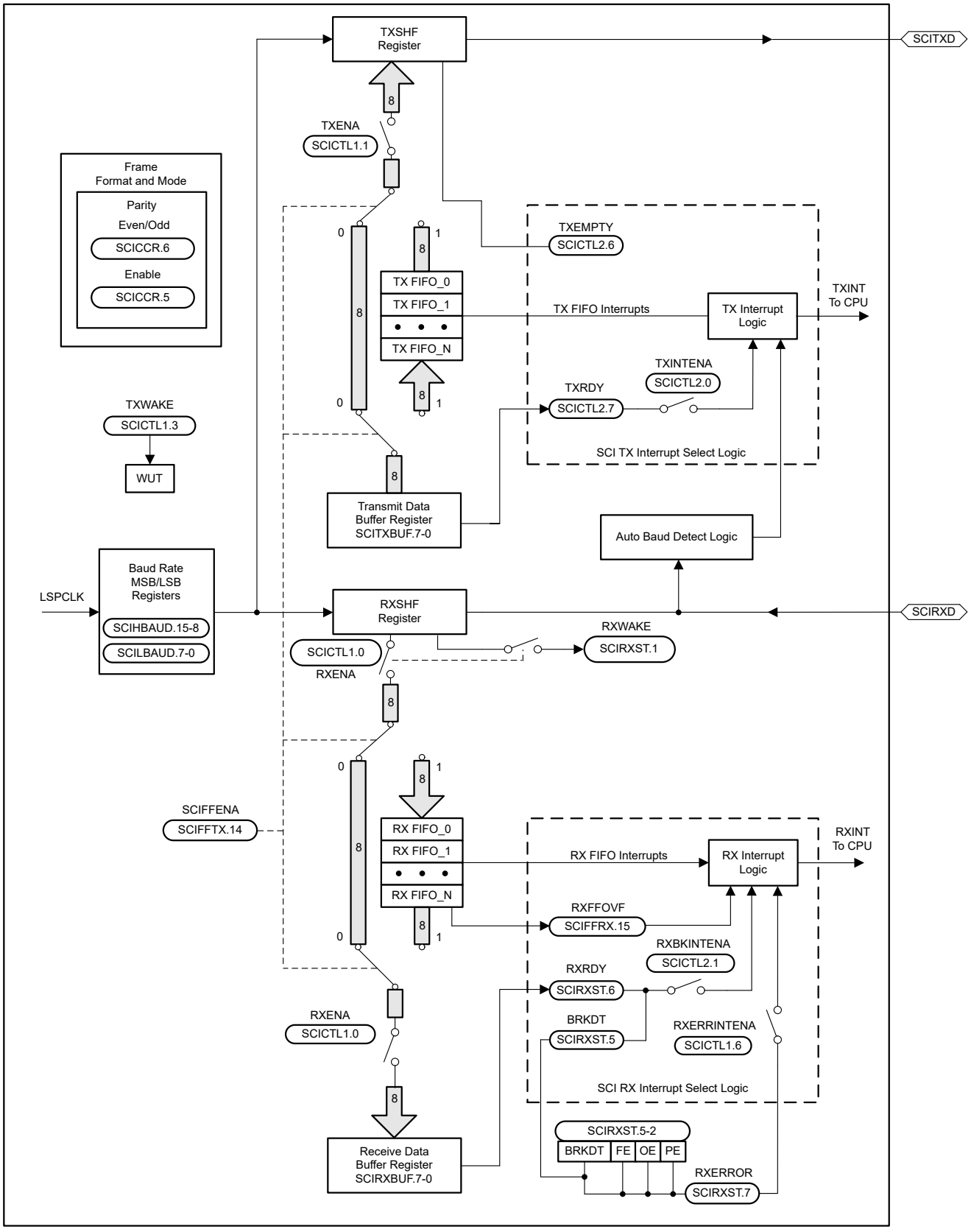



**Figure 19-2. Serial Communications Interface (SCI) Module Block Diagram**



Copyright © 2024 Texas Instruments Incorporated

## 19.4 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.

## 19.5 Multiprocessor and Asynchronous Communication Modes

The SCI has two multiprocessor protocols, the idle-line multiprocessor mode (see Section 19.8) and the address-bit multiprocessor mode (see Section 19.9). 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 19.10) 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



Copyright © 2024 Texas Instruments Incorporated

## 19.6 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 19-3, 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 19-3.

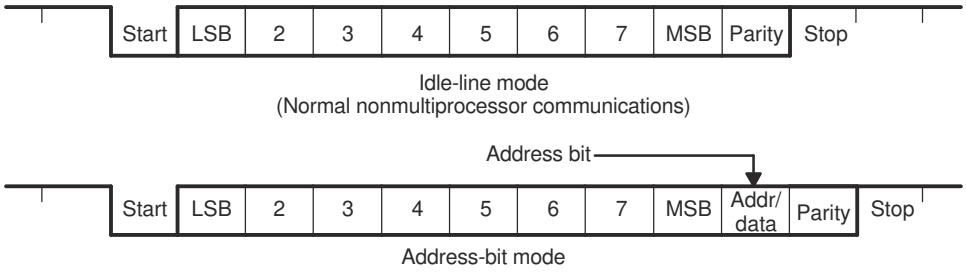



**Figure 19-3. Typical SCI Data Frame Formats**

To program the data format, use the SCICCR register. The bits used to program the data format are shown in Table 19-2.

**Table 19-2. Programming the Data Format Using SCICCR**

| Bits | Bit Name | Designation | Functions |
|---|---|---|---|
| 2-0 | SCICHAR | 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. |



Copyright © 2024 Texas Instruments Incorporated

## 19.7 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.

### *19.7.1 Recognizing the Address Byte*

A processor recognizes an address byte differently, depending on the multiprocessor mode used. For example:

- The idle-line mode (Section 19.8) 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 19.9) 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.

### *19.7.2 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.

### *19.7.3 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.



Copyright © 2024 Texas Instruments Incorporated

## 19.8 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 19-4 (ADDR/IDLE MODE bit is bit 3 of SCICCR).

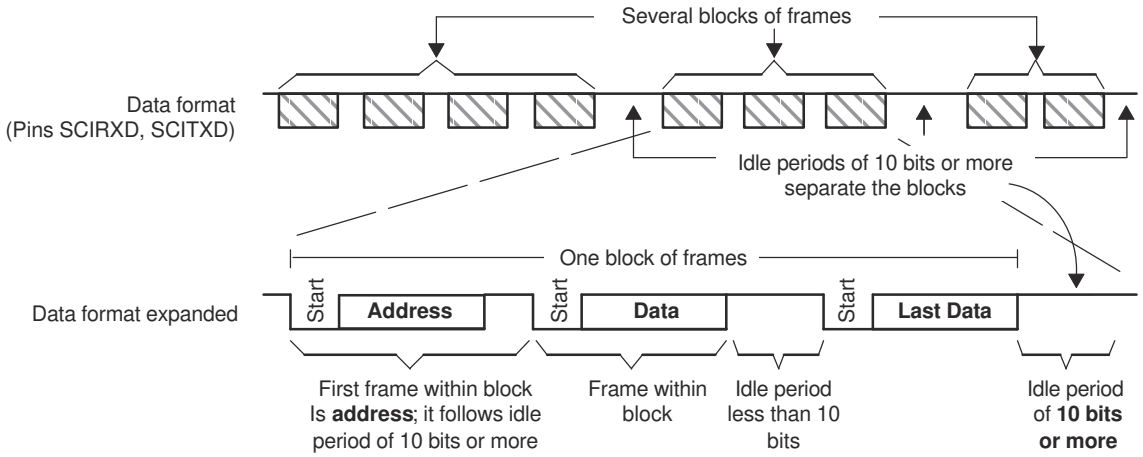



**Figure 19-4. Idle-Line Multiprocessor Communication Format**

### *19.8.1 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.



Copyright © 2024 Texas Instruments Incorporated

### 19.8.2 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.)

### 19.8.3 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 19-5.

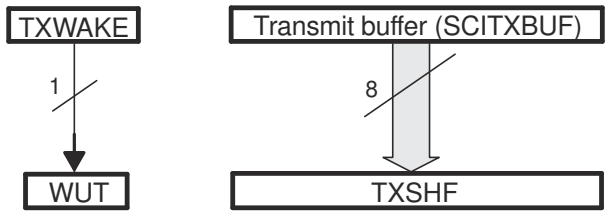



**Figure 19-5. Double-Buffered WUT and TXSHF**

#### 19.8.3.1 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.

### 19.8.4 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.



Copyright © 2024 Texas Instruments Incorporated

## 19.9 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 19-6).

### *19.9.1 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.

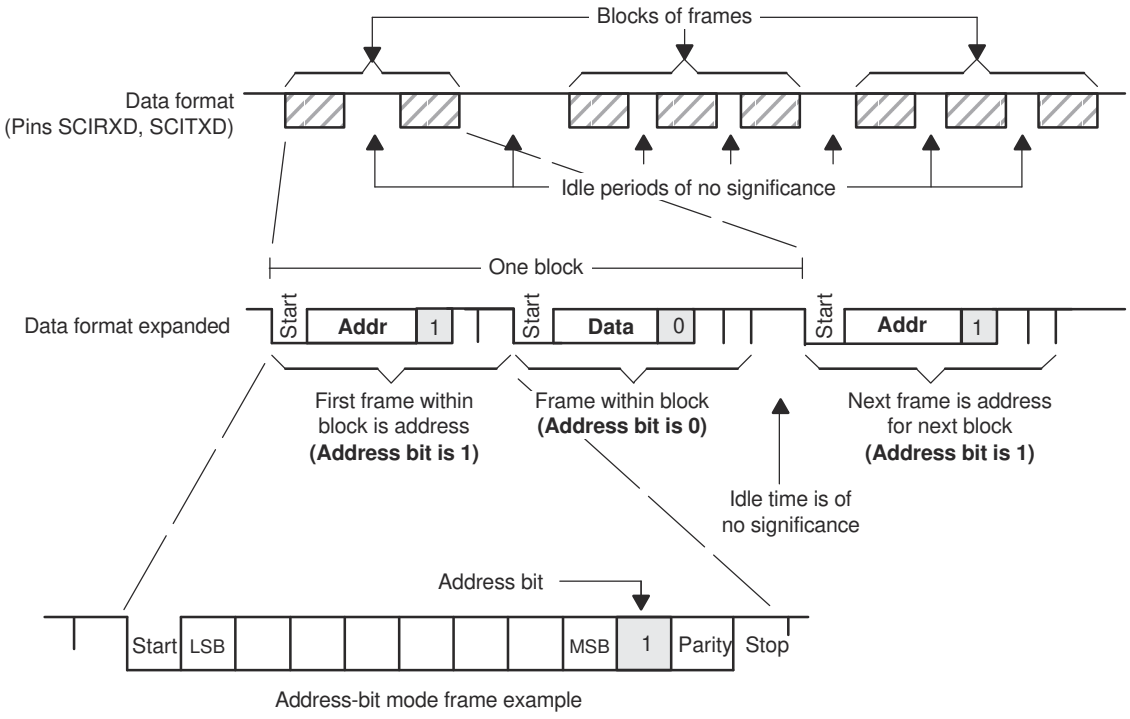



**Figure 19-6. Address-Bit Multiprocessor Communication Format**



Copyright © 2024 Texas Instruments Incorporated

## 19.10 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 19-7). 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 19-7. 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 19-7 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.

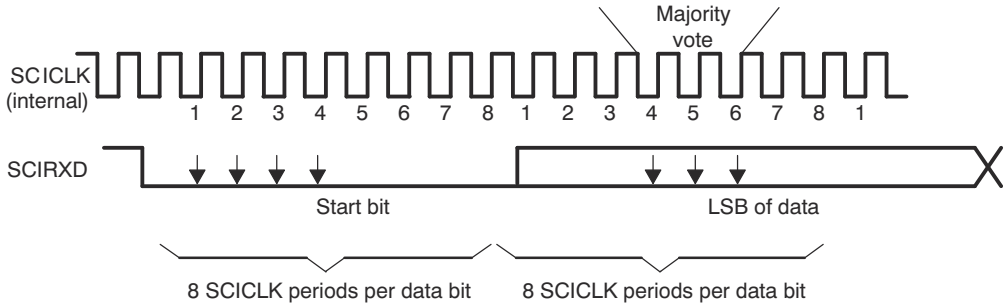



**Figure 19-7. SCI Asynchronous Communications Format**



Copyright © 2024 Texas Instruments Incorporated

### 19.10.1 Receiver Signals in Communication Modes

Figure 19-8 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

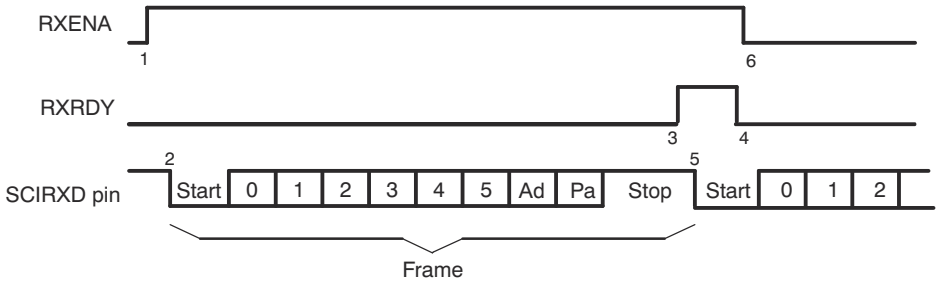



**Figure 19-8. SCI RX Signals in Communication Modes**

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.



Copyright © 2024 Texas Instruments Incorporated

### 19.10.2 Transmitter Signals in Communication Modes

Figure 19-9 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

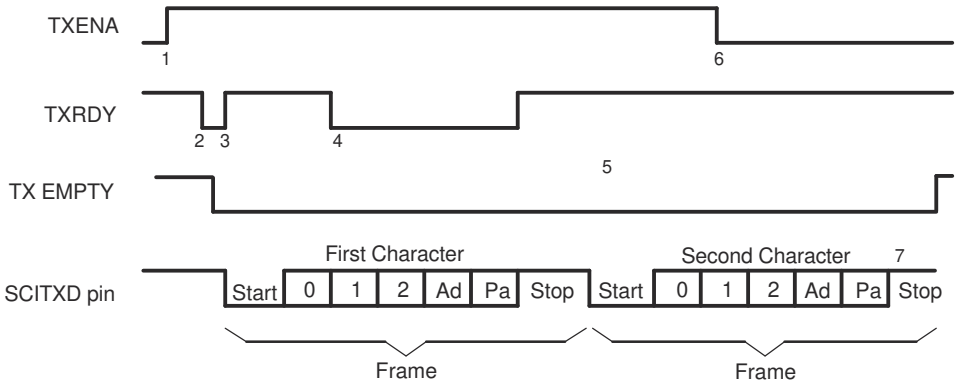



**Figure 19-9. SCI TX Signals in Communications Mode**

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.



Copyright © 2024 Texas Instruments Incorporated

## 19.11 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**

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).



Copyright © 2024 Texas Instruments Incorporated

### 19.11.1 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

## 19.12 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 19-3 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 19-3. Asynchronous Baud Register Values for Common SCI Bit Rates**

| | LSPCLK Clock Frequency, 100MHz | | |
|---|---|---|---|
| Baud Rate | 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 |



Copyright © 2024 Texas Instruments Incorporated

## 19.13 SCI Enhanced Features

The C28x SCI features autobaud detection and transmit/receive FIFO. The following section explains the FIFO operation.

### *19.13.1 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 19-10 and Table 19-4 explain the operation/configuration of SCI interrupts in nonFIFO/FFO mode.



Copyright © 2024 Texas Instruments Incorporated

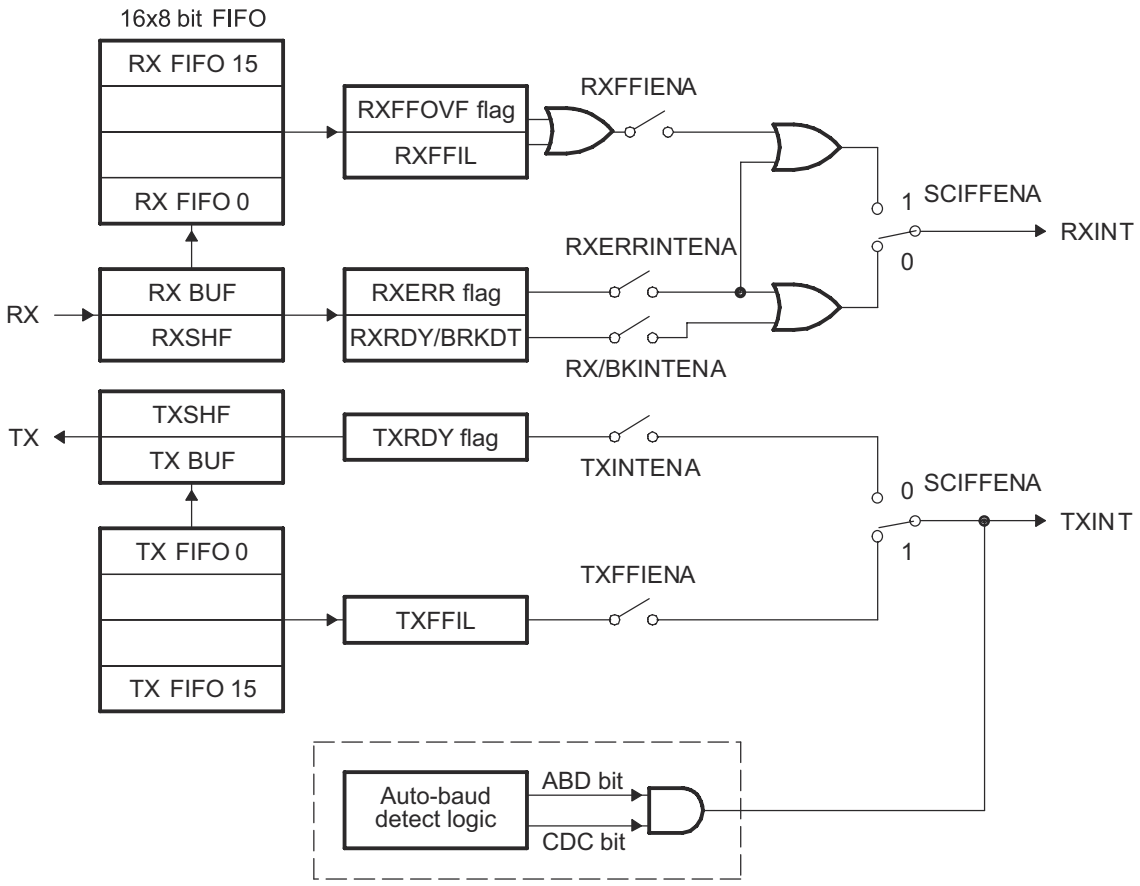



**Figure 19-10. SCI FIFO Interrupt Flags and Enable Logic**

**Table 19-4. SCI Interrupt Flags**

| FIFO Options[1] | SCI Interrupt Source | Interrupt Flags | Interrupt Enables | FIFO Enable SCIFFENA | Interrupt Line |
|---|---|---|---|---|---|
| SCI without FIFO | Receive error | RXERR[2] | RXERRINTENA | 0 | RXINT |
| | Receive break | BRKDT | RX/BKINTENA | 0 | RXINT |
| | Data receive | RXRDY | RX/BKINTENA | 0 | RXINT |
| | Transmit empty | TXRDY | TXINTENA | 0 | TXINT |
| SCI with FIFO | Receive error and receive break | RXERR | RXERRINTENA | 1 | RXINT |
| | FIFO receive | RXFFIL | RXFFIENA | 1 | RXINT |
| | Transmit empty | TXFFIL | TXFFIENA | 1 | TXINT |
| Auto-baud | Auto-baud detected | ABD | Don't care | x | TXINT |

(1) FIFO mode TXSHF is directly loaded after delay value, TXBUF is not used.
(2) RXERR can be set by BRKDT, FE, OE, PE flags. In FIFO mode, BRKDT interrupt is only through RXERR flag.



Copyright © 2024 Texas Instruments Incorporated

### 19.13.2 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.

### 19.13.3 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.



Copyright © 2024 Texas Instruments Incorporated

## 19.14 Software

### 19.14.1 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 C2000Ware Examples.

## 19.15 SCI Registers

The section describes the Serial Communication Interface module registers.

### 19.15.1 SCI Base Addresses

**Table 19-5. SCI Base Address Table**

| Device Registers | Register Name | Start Address | End Address |
|---|---|---|---|
| SciaRegs | SCI_REGS | 0x0000_7200 | 0x0000_720F |
| ScibRegs | SCI_REGS | 0x0000_7210 | 0x0000_721F |
| ScicRegs | SCI_REGS | 0x0000_7220 | 0x0000_722F |
| ScidRegs | SCI_REGS | 0x0000_7230 | 0x0000_723F |



Copyright © 2024 Texas Instruments Incorporated

### 19.15.2 SCI_REGS Registers

Table 19-6 lists the memory-mapped registers for the SCI_REGS registers. All register offset addresses not listed in Table 19-6 should be considered as reserved locations and the register contents should not be modified.

**Table 19-6. SCI_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | SCICCR | Communications control register | | Go |
| 1h | SCICTL1 | Control register 1 | | Go |
| 2h | SCIHBAUD | Baud rate (high) register | | Go |
| 3h | SCILBAUD | Baud rate (low) register | | Go |
| 4h | SCICTL2 | Control register 2 | | Go |
| 5h | SCIRXST | Receive status register | | Go |
| 6h | SCIRXEMU | Receive emulation buffer register | | Go |
| 7h | SCIRXBUF | Receive data buffer | | Go |
| 9h | SCITXBUF | Transmit data buffer | | Go |
| Ah | SCIFFTX | FIFO transmit register | | Go |
| Bh | SCIFFRX | FIFO receive register | | Go |
| Ch | SCIFFCT | FIFO control register | | Go |
| Fh | SCIPRI | SCI priority control | | Go |

Complex bit access types are encoded to fit into small table cells. Table 19-7 shows the codes that are used for access types in this section.

**Table 19-7. SCI_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| R-0 | R<br>-0 | Read<br>Returns 0s |
| Write Type | | |
| W | W | Write |
| W1S | W<br>1S | Write<br>1 to set |
| Reset or Default Value | | |
| *-n* | | Value after reset or the default value |



Copyright © 2024 Texas Instruments Incorporated

### 19.15.2.1 SCICCR Register (Offset = 0h) [Reset = 0000h]

SCICCR is shown in Figure 19-11 and described in Table 19-8.

Return to the Summary Table.

SCICCR defines the character format, protocol, and communications mode used by the SCI.

**Figure 19-11. 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 19-8. 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.<br>This bit specifies the number of stop bits transmitted. The receiver checks for only one stop bit.<br>Reset type: SYSRSn<br>0h (R/W) = One stop bit<br>1h (R/W) = Two stop bits |
| 6 | PARITY | R/W | 0h | SCI parity odd/even selection.<br>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).<br>Reset type: SYSRSn<br>0h (R/W) = Odd parity<br>1h (R/W) = Even parity |
| 5 | PARITYENA | R/W | 0h | SCI parity enable.<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = Parity disabled<br>no parity bit is generated during transmission or is expected during reception<br>1h (R/W) = Parity is enabled |
| 4 | LOOPBKENA | R/W | 0h | Loop Back test mode enable.<br>This bit enables the Loop Back test mode where the Tx pin is internally connected to the Rx pin.<br>Reset type: SYSRSn<br>0h (R/W) = Loop Back test mode disabled<br>1h (R/W) = Loop Back test mode enabled |



Copyright © 2024 Texas Instruments Incorporated

**Table 19-8. SCICCR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3 | ADDRIDLE_MODE | R/W | 0h | SCI multiprocessor mode control bit.<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = Idle-line mode protocol selected<br>1h (R/W) = Address-bit mode protocol selected |
| 2-0 | SCICHAR | R/W | 0h | Character-length control bits 2-0.<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = SCICHAR_LENGTH_1<br>1h (R/W) = SCICHAR_LENGTH_2<br>2h (R/W) = SCICHAR_LENGTH_3<br>3h (R/W) = SCICHAR_LENGTH_4<br>4h (R/W) = SCICHAR_LENGTH_5<br>5h (R/W) = SCICHAR_LENGTH_6<br>6h (R/W) = SCICHAR_LENGTH_7<br>7h (R/W) = SCICHAR_LENGTH_8 |



Copyright © 2024 Texas Instruments Incorporated

### 19.15.2.2 SCICTL1 Register (Offset = 1h) [Reset = 0000h]

SCICTL1 is shown in Figure 19-12 and described in Table 19-9.

Return to the Summary Table.

SCICTL1 controls the receiver/transmitter enable, TXWAKE and SLEEP functions, and the SCI software reset.

**Figure 19-12. 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 19-9. 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.<br>Setting this bit enables an interrupt if the RX ERROR bit (SCIRXST, bit 7) becomes set because of errors occurring.<br>Reset type: SYSRSn<br>0h (R/W) = Receive error interrupt disabled<br>1h (R/W) = Receive error interrupt enabled |
| 5 | SWRESET | R/W | 0h | SCI software reset (active low).<br>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].<br>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).<br>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:<br>Value After SW SCI Flag Register Bit<br>RESET<br>1 TXRDY SCICTL2, bit 7<br>1 TX EMPTY SCICTL2, bit 6<br>0 RXWAKE SCIRXST, bit 1<br>0 PE SCIRXST, bit 2<br>0 OE SCIRXST, bit 3<br>0 FE SCIRXST, bit 4<br>0 BRKDT SCIRXST, bit 5<br>0 RXRDY SCIRXST, bit 6<br>0 RX ERROR SCIRXST, bit 7<br>Reset type: SYSRSn<br>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.<br>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 |



Copyright © 2024 Texas Instruments Incorporated

**Table 19-9. SCICTL1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3 | TXWAKE | R/W | 0h | SCI transmitter wake-up method select.<br>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)<br>Reset type: SYSRSn<br>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<br>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)<br>it is cleared by a system reset or the transfer of TXWAKE to the WUT flag. |
| 2 | SLEEP | R/W | 0h | SCI sleep.<br>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.<br>The receiver still operates when the SLEEP bit is set<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = Sleep mode disabled<br>1h (R/W) = Sleep mode enabled |
| 1 | TXENA | R/W | 0h | SCI transmitter enable.<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = Transmitter disabled<br>1h (R/W) = Transmitter enabled |
| 0 | RXENA | R/W | 0h | SCI receiver enable.<br>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).<br>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.<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = Prevent received characters from transfer into the SCIRXEMU and SCIRXBUF receiver buffers<br>1h (R/W) = Send received characters to SCIRXEMU and SCIRXBUF |



Copyright © 2024 Texas Instruments Incorporated

### 19.15.2.3 SCIHBAUD Register (Offset = 2h) [Reset = 0000h]

SCIHBAUD is shown in Figure 19-13 and described in Table 19-10.

Return to the Summary Table.

The values in SCIHBAUD and SCILBAUD specify the baud rate for the SCI.

**Figure 19-13. 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 19-10. 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).<br>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.<br>BRR = (SCIHBAUD << 8) + (SCILBAUD)<br>The SCI baud rate is calculated using the following equation:<br>SCI Asynchronous Baud = LSPCLK / ((BRR + 1) *8)<br>Alternatively,<br>BRR = LSPCLK / (SCI Asynchronous Baud * 8) - 1<br>Note that the above formulas are applicable only when 0 < BRR < 65536. If BRR = 0, then<br>SCI Asynchronous Baud = LSPCLK / 16<br>Where: BRR = the 16-bit value (in decimal) in the baud-select registers<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 19.15.2.4 SCILBAUD Register (Offset = 3h) [Reset = 0000h]

SCILBAUD is shown in Figure 19-14 and described in Table 19-11.

Return to the Summary Table.

The values in SCIHBAUD and SCILBAUD specify the baud rate for the SCI.

**Figure 19-14. 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 19-11. 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<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 19.15.2.5 SCICTL2 Register (Offset = 4h) [Reset = 00C0h]

SCICTL2 is shown in Figure 19-15 and described in Table 19-12.

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 19-15. 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 19-12. SCICTL2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R | 0h | Reserved |
| 7 | TXRDY | R | 1h | Transmitter buffer register ready flag.<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = SCITXBUF is full<br>1h (R/W) = SCITXBUF is ready to receive the next character |
| 6 | TXEMPTY | R | 1h | Transmitter empty flag.<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = Transmitter buffer or shift register or both are loaded with data<br>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.<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = Disable RXRDY/BRKDT interrupt<br>1h (R/W) = Enable RXRDY/BRKDT interrupt |



Copyright © 2024 Texas Instruments Incorporated

**Table 19-12. SCICTL2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | TXINTENA | R/W | 0h | SCITXBUF-register interrupt enable.<br>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).<br>0 Disable TXRDY interrupt<br>1 Enable TXRDY interrupt.<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = Disable TXRDY interrupt<br>1h (R/W) = Enable TXRDY interrupt |



Copyright © 2024 Texas Instruments Incorporated

### 19.15.2.6 SCIRXST Register (Offset = 5h) [Reset = 0000h]

SCIRXST is shown in Figure 19-16 and described in Table 19-13.

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 19-16. 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 19-13. SCIRXST Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R | 0h | Reserved |
| 7 | RXERROR | R | 0h | SCI receiver error flag.<br>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).<br>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<br>it is cleared by an active SW RESET, channel reset (SCIRST), or by a system reset.<br>Reset type: SYSRSn<br>0h (R/W) = No error flags set<br>1h (R/W) = Error flag(s) set |
| 6 | RXRDY | R | 0h | SCI receiver-ready flag.<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = No new character in SCIRXBUF<br>1h (R/W) = Character ready to be read from SCIRXBUF |



Copyright © 2024 Texas Instruments Incorporated

**Table 19-13. SCIRXST Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 5 | BRKDT | R | 0h | SCI break-detect flag.<br>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.<br>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 && PE==1 && 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).<br>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.<br>A BRKDT interrupt can occur even if the receiver SLEEP bit is set to 1.<br>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.<br>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.<br>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.<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = No break condition<br>1h (R/W) = Break condition occurred |
| 4 | FE | R | 0h | SCI framing-error flag.<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = No framing error detected<br>1h (R/W) = Framing error detected |



Copyright © 2024 Texas Instruments Incorporated

**Table 19-13. SCIRXST Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3 | OE | R | 0h | SCI overrun-error flag.<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = No overrun error detected<br>1h (R/W) = Overrun error detected |
| 2 | PE | R | 0h | SCI parity-error flag.<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = No parity error or parity is disabled<br>1h (R/W) = Parity error is detected |
| 1 | RXWAKE | R | 0h | Receiver wake-up-detect flag<br>Reset type: SYSRSn<br>0h (R/W) = No detection of a receiver wake-up condition<br>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:<br>- The transfer of the first byte after the address byte to SCIRXBUF (only in non-FIFO mode)<br>- The reading of SCIRXBUF<br>- An active SW RESET<br>- Channel reset (SCIRST)<br>- A system reset |
| 0 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 19.15.2.7 SCIRXEMU Register (Offset = 6h) [Reset = 0000h]

SCIRXEMU is shown in Figure 19-17 and described in Table 19-14.

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 19-17. 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 19-14. SCIRXEMU Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R | 0h | Reserved |
| 7-0 | ERXDT | R | 0h | Receive emulation buffer data<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 19.15.2.8 SCIRXBUF Register (Offset = 7h) [Reset = 0000h]

SCIRXBUF is shown in Figure 19-18 and described in Table 19-15.

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 19-18. 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 19-15. 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)<br>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.<br>Reset type: SYSRSn<br>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.<br>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)<br>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.<br>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.<br>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).<br>Reset type: SYSRSn<br>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.<br>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 |



Copyright © 2024 Texas Instruments Incorporated

**Table 19-15. SCIRXBUF Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 7-0 | SAR | R | 0h | Receive Character bits<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 19.15.2.9 SCITXBUF Register (Offset = 9h) [Reset = 0000h]

SCITXBUF is shown in Figure 19-19 and described in Table 19-16.

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 19-19. 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 19-16. SCITXBUF Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R | 0h | Reserved |
| 7-0 | TXDT | R/W | 0h | Transmit data buffer<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 19.15.2.10 SCIFFTX Register (Offset = Ah) [Reset = A000h]

SCIFFTX is shown in Figure 19-20 and described in Table 19-17.

Return to the Summary Table.

SCIFFTX controls the transmit FIFO interrupt, FIFO enhancements, and reset for the SCI transmit and receive channels.

**Figure 19-20. 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 19-17. SCIFFTX Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | SCIRST | R/W | 1h | SCI Reset<br>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.<br>1 SCI FIFO can resume transmit or receive. SCIRST should be 1 even for Autobaud logic to work.<br>Reset type: SYSRSn |
| 14 | SCIFFENA | R/W | 0h | SCI FIFO enable<br>Reset type: SYSRSn<br>0h (R/W) = SCI FIFO enhancements are disabled<br>1h (R/W) = SCI FIFO enhancements are enabled |
| 13 | TXFIFORESET | R/W | 1h | Transmit FIFO reset<br>Reset type: SYSRSn<br>0h (R/W) = Reset the FIFO pointer to zero and hold in reset<br>1h (R/W) = Re-enable transmit FIFO operation |
| 12-8 | TXFFST | R | 0h | FIFO status<br>Reset type: SYSRSn<br>0h (R/W) = Transmit FIFO is empty<br>1h (R/W) = Transmit FIFO has 1 words<br>2h (R/W) = Transmit FIFO has 2 words<br>3h (R/W) = Transmit FIFO has 3 words<br>4h (R/W) = Transmit FIFO has 4 words<br>5h (R/W) = Transmit FIFO has 5 words<br>6h (R/W) = Transmit FIFO has 6 words<br>7h (R/W) = Transmit FIFO has 7 words<br>8h (R/W) = Transmit FIFO has 8 words<br>9h (R/W) = Transmit FIFO has 9 words<br>Ah (R/W) = Transmit FIFO has 10 words<br>Bh (R/W) = Transmit FIFO has 11 words<br>Ch (R/W) = Transmit FIFO has 12 words<br>Dh (R/W) = Transmit FIFO has 13 words<br>Eh (R/W) = Transmit FIFO has 14 words<br>Fh (R/W) = Transmit FIFO has 15 words<br>10h (R/W) = Transmit FIFO has 16 words |
| 7 | TXFFINT | R | 0h | Transmit FIFO interrupt<br>Reset type: SYSRSn<br>0h (R/W) = TXFIFO interrupt has not occurred, read-only bit<br>1h (R/W) = TXFIFO interrupt has occurred, read-only bit |



Copyright © 2024 Texas Instruments Incorporated

**Table 19-17. SCIFFTX Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 6 | TXFFINTCLR | R-0/W1S | 0h | Transmit FIFO clear<br>Reset type: SYSRSn<br>0h (R/W) = Write 0 has no effect on TXFIFINT flag bit, Bit reads back a zero<br>1h (R/W) = Write 1 to clear TXFFINT flag in bit 7 |
| 5 | TXFFIENA | R/W | 0h | Transmit FIFO interrrupt enable<br>Reset type: SYSRSn<br>0h (R/W) = TX FIFO interrupt is disabled<br>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.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 19.15.2.11 SCIFFRX Register (Offset = Bh) [Reset = 201Fh]

SCIFFRX is shown in Figure 19-21 and described in Table 19-18.

Return to the Summary Table.

SCIFFRX controls the receive FIFO interrupt, receive FIFO reset, and status of the receive FIFO overflow.

**Figure 19-21. 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 19-18. SCIFFRX Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RXFFOVF | R | 0h | Receive FIFO overflow.<br>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.<br>This bit is cleared by RXFFOVRCLR, a channel reset (SCIRST), or a system reset.<br>Reset type: SYSRSn<br>0h (R/W) = Receive FIFO has not overflowed, read-only bit<br>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<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = Write 0 has no effect on RXFFOVF flag bit, Bit reads back a zero<br>1h (R/W) = Write 1 to clear RXFFOVF flag in bit 15 |
| 13 | RXFIFORESET | R/W | 1h | Receive FIFO reset<br>Reset type: SYSRSn<br>0h (R/W) = Write 0 to reset the FIFO pointer to zero, and hold in reset.<br>1h (R/W) = Re-enable receive FIFO operation |



Copyright © 2024 Texas Instruments Incorporated

**Table 19-18. SCIFFRX Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 12-8 | RXFFST | R | 0h | FIFO status<br>Reset type: SYSRSn<br>0h (R/W) = Receive FIFO is empty<br>1h (R/W) = Receive FIFO has 1 words<br>2h (R/W) = Receive FIFO has 2 words<br>3h (R/W) = Receive FIFO has 3 words<br>4h (R/W) = Receive FIFO has 4 words<br>5h (R/W) = Receive FIFO has 5 words<br>6h (R/W) = Receive FIFO has 6 words<br>7h (R/W) = Receive FIFO has 7 words<br>8h (R/W) = Receive FIFO has 8 words<br>9h (R/W) = Receive FIFO has 9 words<br>Ah (R/W) = Receive FIFO has 10 words<br>Bh (R/W) = Receive FIFO has 11 words<br>Ch (R/W) = Receive FIFO has 12 words<br>Dh (R/W) = Receive FIFO has 13 words<br>Eh (R/W) = Receive FIFO has 14 words<br>Fh (R/W) = Receive FIFO has 15 words<br>10h (R/W) = Receive FIFO has 16 words |
| 7 | RXFFINT | R | 0h | Receive FIFO interrupt<br>Reset type: SYSRSn<br>0h (R/W) = RXFIFO interrupt has not occurred, read-only bit<br>1h (R/W) = RXFIFO interrupt has occurred, read-only bit |
| 6 | RXFFINTCLR | W | 0h | Receive FIFO interrupt clear<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = Write 0 has no effect on RXFIFINT flag bit. Bit reads back a zero.<br>1h (R/W) = Write 1 to clear RXFFINT flag in bit 7 |
| 5 | RXFFIENA | R/W | 0h | Receive FIFO interrupt enable<br>Reset type: SYSRSn<br>0h (R/W) = RX FIFO interrupt is disabled<br>1h (R/W) = RX FIFO interrupt is enabled. This interrupt is triggered whenever the receive FIFO status (RXFFST) bits match (equal to or greater than) the interrupt trigger level bits RXFFIL (bits 4-0). |
| 4-0 | RXFFIL | R/W | 1Fh | Receive FIFO interrupt level bits<br>The receive FIFO generates an interrupt whenever the FIFO status bits (RXFFST4-0) are greater than or equal to the FIFO level bits (RXFFIL4-0). The maximum value that can be assigned to these bits to generate an interrupt cannot be more than the depth of the RX FIFO. The default value of these bits after reset is 11111b. Users should set RXFFIL to best fit their application needs by weighing between the CPU overhead to service the ISR and the best possible usage of received SCI data.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 19.15.2.12 SCIFFCT Register (Offset = Ch) [Reset = 0000h]

SCIFFCT is shown in Figure 19-22 and described in Table 19-19.

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 19-22. 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 19-19. SCIFFCT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | ABD | R | 0h | Auto-baud detect (ABD) bit<br>Reset type: SYSRSn<br>0h (R/W) = Auto-baud detection is not complete. 'A','a' character has not been received successfully.<br>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<br>Reset type: SYSRSn<br>0h (R/W) = Write 0 has no effect on ABD flag bit. Bit reads back a zero.<br>1h (R/W) = Write 1 to clear ABD flag in bit 15. |
| 13 | CDC | R/W | 0h | CDC calibrate A-detect bit<br>Reset type: SYSRSn<br>0h (R/W) = Disables auto-baud alignment<br>1h (R/W) = Enables auto-baud alignment |
| 12-8 | RESERVED | R | 0h | Reserved |
| 7-0 | FFTXDLY | R/W | 0h | FIFO transfer delay. These bits define the delay between every transfer from FIFO transmit bufferto transmit shift register. The delay is defined in the number of SCI serial baud clock cycles. The 8 bit register could define a minimum delay of 0 baud clock cycles and a maximum of 256 baud clock cycles<br>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.<br>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.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 19.15.2.13 SCIPRI Register (Offset = Fh) [Reset = 0000h]

SCIPRI is shown in Figure 19-23 and described in Table 19-20.

Return to the Summary Table.

SCIPRI determines what happens when an emulation suspend event occurs.

**Figure 19-23. 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 19-20. 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.<br>Reset type: SYSRSn<br>0h (R/W) = Immediate stop on suspend<br>1h (R/W) = Complete current receive/transmit sequence before stopping<br>2h (R/W) = Free run<br>3h (R/W) = Free run |
| 2-0 | RESERVED | R | 0h | Reserved |

### *19.15.3 SCI Registers to Driverlib Functions*

**Table 19-21. SCI Registers to Driverlib Functions**

| File | Driverlib Function |
|---|---|
| **SCICCR** | |
| 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 |
| **SCICTL1** | |
| 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 |



Copyright © 2024 Texas Instruments Incorporated

**Table 19-21. 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 |
| **SCIHBAUD** | |
| sci.c | SCI_setConfig |
| sci.c | SCI_setBaud |
| sci.h | SCI_lockAutobaud |
| sci.h | SCI_getConfig |
| **SCILBAUD** | |
| sci.c | SCI_setConfig |
| sci.c | SCI_setBaud |
| sci.h | SCI_lockAutobaud |
| sci.h | SCI_getConfig |
| **SCICTL2** | |
| sci.c | SCI_enableInterrupt |
| sci.c | SCI_disableInterrupt |
| sci.c | SCI_getInterruptStatus |
| sci.h | SCI_isSpaceAvailableNonFIFO |
| sci.h | SCI_isTransmitterBusy |
| **SCIRXST** | |
| sci.c | SCI_getInterruptStatus |
| sci.h | SCI_isDataAvailableNonFIFO |
| sci.h | SCI_getRxStatus |
| **SCIRXEMU** | |
| - | |
| **SCIRXBUF** | |
| sci.c | SCI_readCharArray |
| sci.h | SCI_readCharBlockingFIFO |
| sci.h | SCI_readCharBlockingNonFIFO |
| sci.h | SCI_readCharNonBlocking |
| **SCITXBUF** | |
| sci.c | SCI_writeCharArray |
| sci.h | SCI_writeCharBlockingFIFO |
| sci.h | SCI_writeCharBlockingNonFIFO |
| sci.h | SCI_writeCharNonBlocking |
| **SCIFFTX** | |
| 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 |



Copyright © 2024 Texas Instruments Incorporated

**Table 19-21. 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 |
| **SCIFFRX** | |
| 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 |
| **SCIFFCT** | |
| sci.h | SCI_lockAutobaud |
| **SCIPRI** | |
| - | |



Copyright © 2024 Texas Instruments Incorporated

Chapter 20

# 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, I2CMDR.


**20.1 Introduction** ..... 2342
**20.2 Configuring Device Pins** ..... 2347
**20.3 I2C Module Operational Details** ..... 2347
**20.4 Interrupt Requests Generated by the I2C Module** ..... 2358
**20.5 Resetting or Disabling the I2C Module** ..... 2362
**20.6 Software** ..... 2363
**20.7 I2C Registers** ..... 2365





Copyright © 2024 Texas Instruments Incorporated

## 20.1 Introduction

The I2C module supports any slave or master I2C-compatible device. Figure 20-1 shows an example of multiple I2C modules connected for a two-way transfer from one device to other devices.

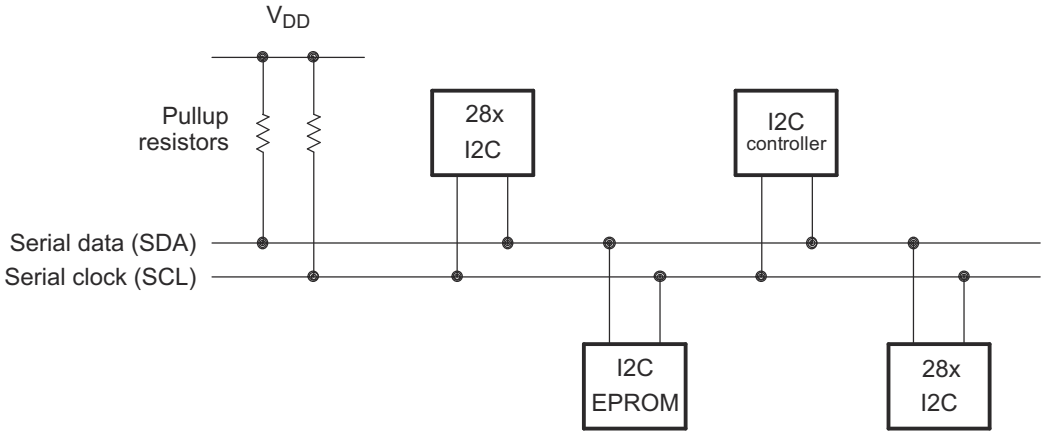



**Figure 20-1. Multiple I2C Modules Connected**

### *20.1.1 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*



Copyright © 2024 Texas Instruments Incorporated

### 20.1.2 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

### 20.1.3 Features Not Supported

The I2C module does not support:

- High-speed mode (Hs-mode)
- CBUS-compatibility mode



Copyright © 2024 Texas Instruments Incorporated

### 20.1.4 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 20-2. 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 20-2 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.



Copyright © 2024 Texas Instruments Incorporated

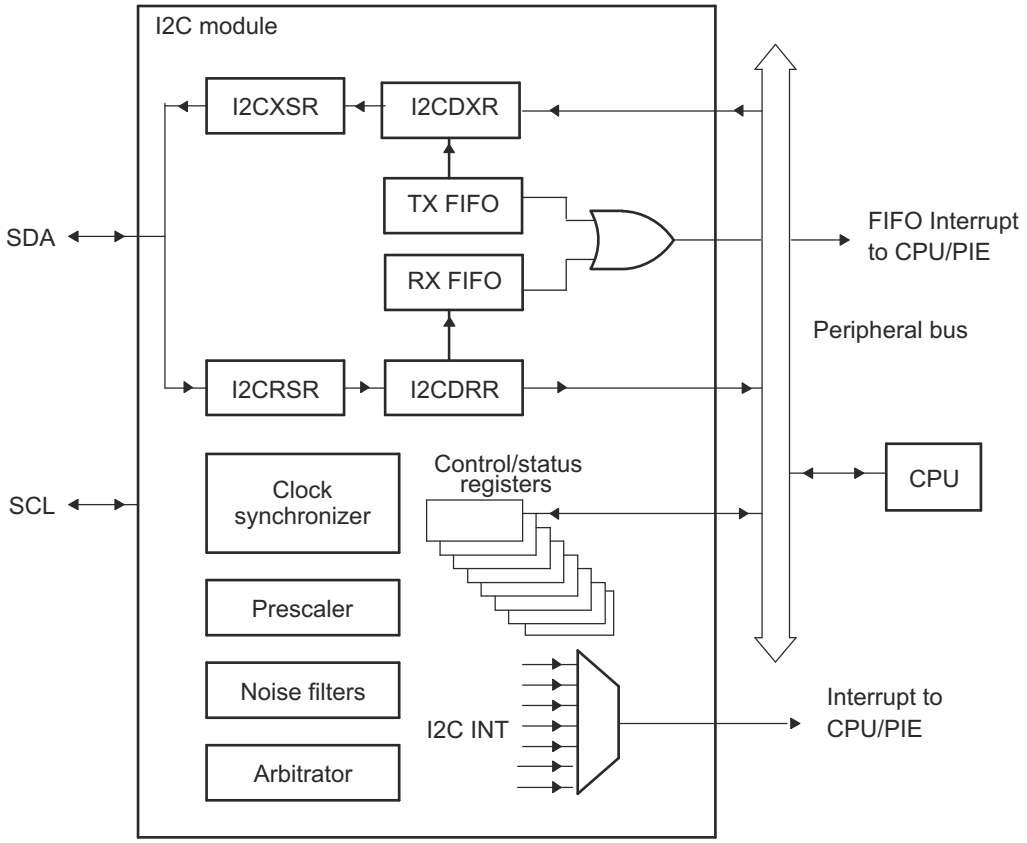



**Figure 20-2. I2C Module Conceptual Block Diagram**

### *20.1.5 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 20-3 shows the clock generation diagram for I2C module.

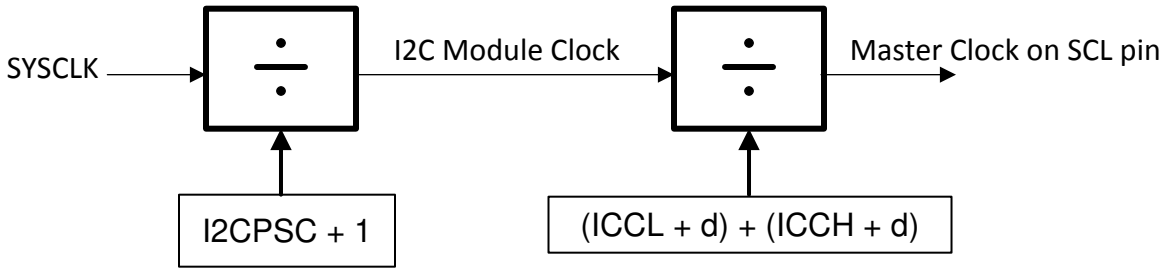



**Figure 20-3. Clocking Diagram for the I2C Module**

**Note**

To meet all of the I2C protocol timing specifications, the I2C module clock must be between 7 to 12MHz.



Copyright © 2024 Texas Instruments Incorporated

To specify the divide-down value, initialize the IPSC field of the prescaler register, I2CPSC. The resulting frequency is:

$$\text{I2C Module Clock (Fmod)} = \frac{\text{SYSCLK}}{(\text{I2CPSC} + 1)} \quad (28)$$

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 20-3, 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 20.1.6 for the master clock frequency equation.

### *20.1.6 I2C Clock Divider Registers (I2CCLKL and I2CCLKH)*

As explained in Section 20.1.5, 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 20-4, 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 I2CCKLH. For each master clock cycle, ICCH determines the amount of time the signal is high.

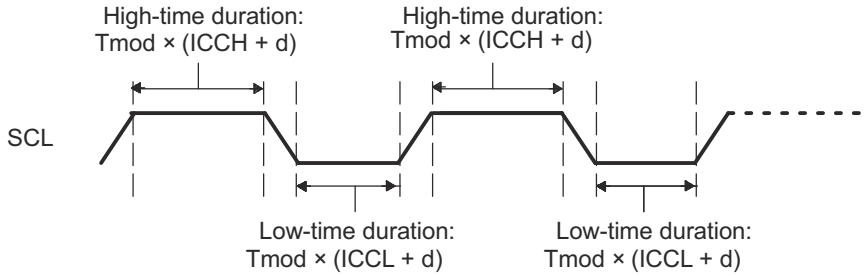



**Figure 20-4. Roles of the Clock Divide-Down Values (ICCL and ICCH)**

#### 20.1.6.1 Formula for the Master Clock Period

The master clock period (Tmst) is a multiple of the period of the I2C Module Clock (Tmod):

$$\text{Master Clock period (Tmst)} = \frac{[\,(\text{ICCH} + d) + (\text{ICCL} + d)\,]}{\text{I2C Module Clock (Fmod)}} \quad (29)$$

where d depends on the divide-down value IPSC, as shown in Table 20-1. IPSC is described in the I2CPSC register.

**Table 20-1. Dependency of Delay d on the Divide-Down Value IPSC**

| IPSC | d |
|---|---|
| 0 | 7 |
| 1 | 6 |
| Greater than 1 | 5 |



Copyright © 2024 Texas Instruments Incorporated

## 20.2 Configuring Device Pins

The GPIO mux registers must be configured to connect this peripheral to the device pins. To avoid glitches on the pins, the GPyGMUX bits must be configured first (while keeping the corresponding GPyMUX bits at the default of zero), followed by writing the GPyMUX register to the desired value.

Some IO functionality is defined by GPIO register settings independent of this peripheral. For input signals, the GPIO input qualification 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.

## 20.3 I2C Module Operational Details

This section provides an overview of the I2C bus protocol and how it is implemented.

### *20.3.1 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.

### *20.3.2 Selecting Pullup Resistors*

The chosen pullup resistor must meet the I2C standard timings. In most circumstances, 2.2kΩ 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*.

### *20.3.3 Data Validity*

The data on SDA must be stable during the high period of the clock (see Figure 20-5). The high or low state of the data line, SDA, must change only when the clock signal on SCL is low.

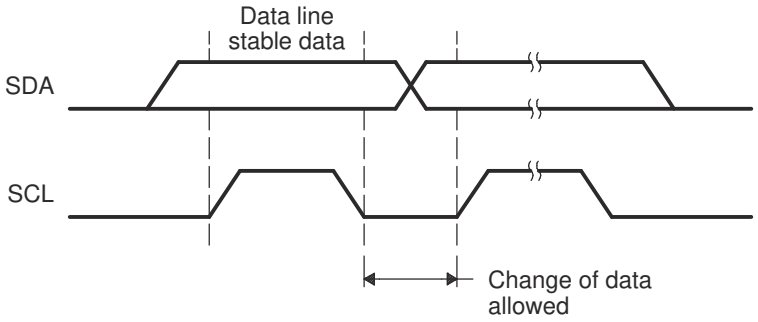



**Figure 20-5. Bit Transfer on the I2C bus**

### *20.3.4 Operating Modes*

The I2C module has four basic operating modes to support data transfers as a master and as a slave. See Table 20-2 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.



Copyright © 2024 Texas Instruments Incorporated

**Table 20-2. Operating Modes of the I2C Module**

| Operating Mode | Description |
|---|---|
| Slave-receiver mode | The I2C module is a slave and receives data from a master.<br>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 Section 20.3.8 for more details. |
| Slave-transmitter mode | The I2C module is a slave and transmits data to a master.<br>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/ $\overline{W}$ = 1. As a slave-transmitter, the I2C module then shifts the serial data out on SDA with the clock pulses that are generated by the master. While a slave, the I2C module does not generate the clock signal, but it can hold SCL low while the intervention of the device is required (XSMT = 0 in I2CSTR) after a byte has been transmitted. See Section 20.3.8 for more details. |
| Master-receiver mode | The I2C module is a master and receives data from a slave.<br>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/ $\overline{W}$ = 1. Serial data bits on SDA are shifted into the I2C module with the clock pulses generated by the I2C module on SCL. The clock pulses are inhibited and SCL is held low when the intervention of the device is required (RSFULL = 1 in I2CSTR) after a byte has been received. |
| Master-transmitter mode | The I2C module is a master and transmits control information and data to a slave.<br>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 20-3. Master-Transmitter/Receiver Bus Activity Defined by the RM, STT, and STP Bits of I2CMDR**

| RM | STT | STP | Bus Activity(1) | Description |
|---|---|---|---|---|
| 0 | 0 | 0 | None | No activity |
| 0 | 0 | 1 | P | STOP condition |
| 0 | 1 | 0 | S-A-D..(n)..D. | START condition, slave address, n data bytes (n = value in I2CCNT) |
| 0 | 1 | 1 | S-A-D..(n)..D-P | START condition, slave address, n data bytes, STOP condition (n = value in I2CCNT) |
| 1 | 0 | 0 | None | No activity |
| 1 | 0 | 1 | P | STOP condition |
| 1 | 1 | 0 | S-A-D-D-D. | Repeat mode transfer: START condition, slave address, continuous data transfers until STOP condition or next START condition |
| 1 | 1 | 1 | None | Reserved bit combination (No activity) |

(1) S = START condition; A = Address; D = Data byte; P = STOP condition;



Copyright © 2024 Texas Instruments Incorporated

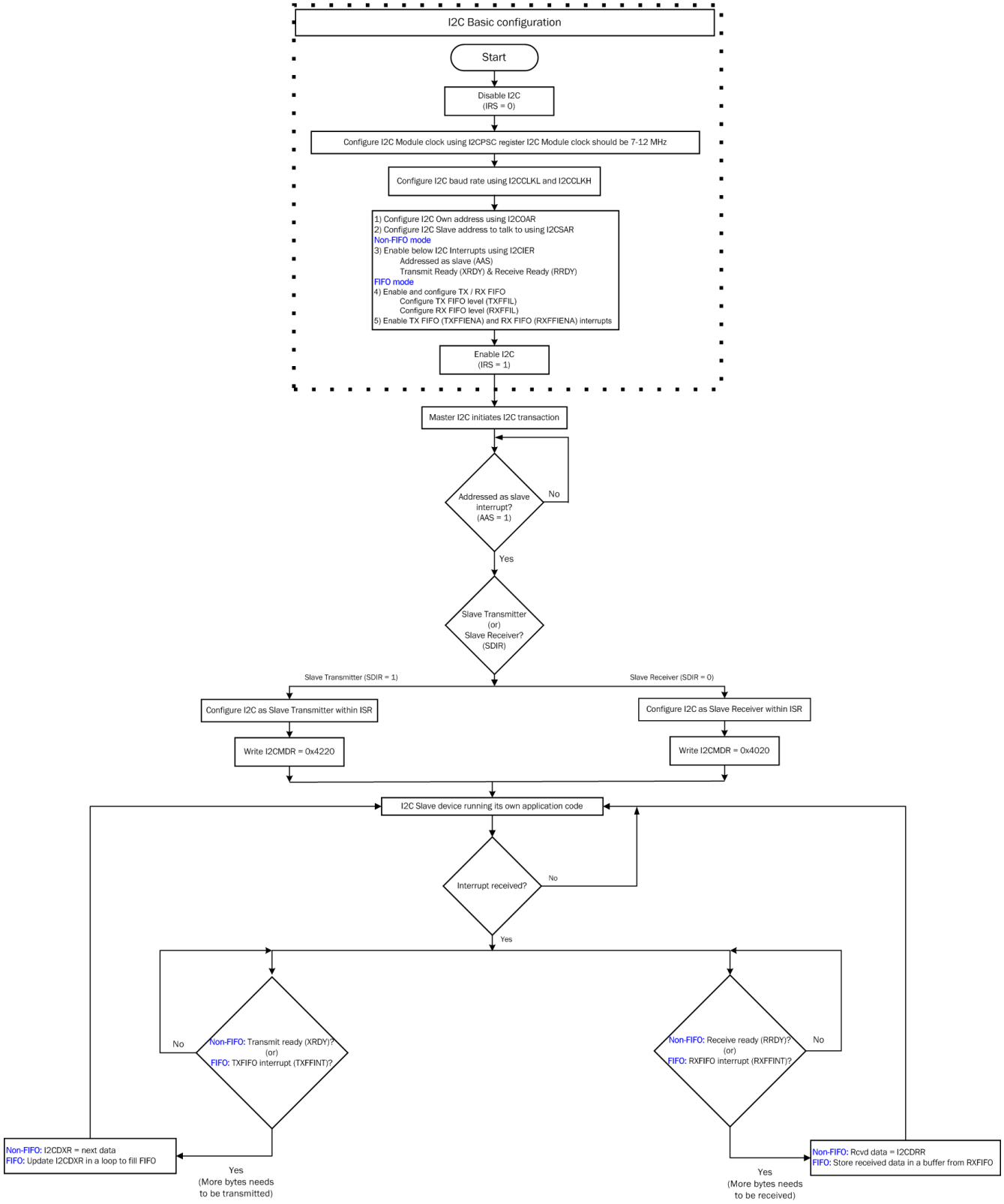



**Figure 20-6. I2C Slave TX / RX Flowchart**


Copyright © 2024 Texas Instruments Incorporated

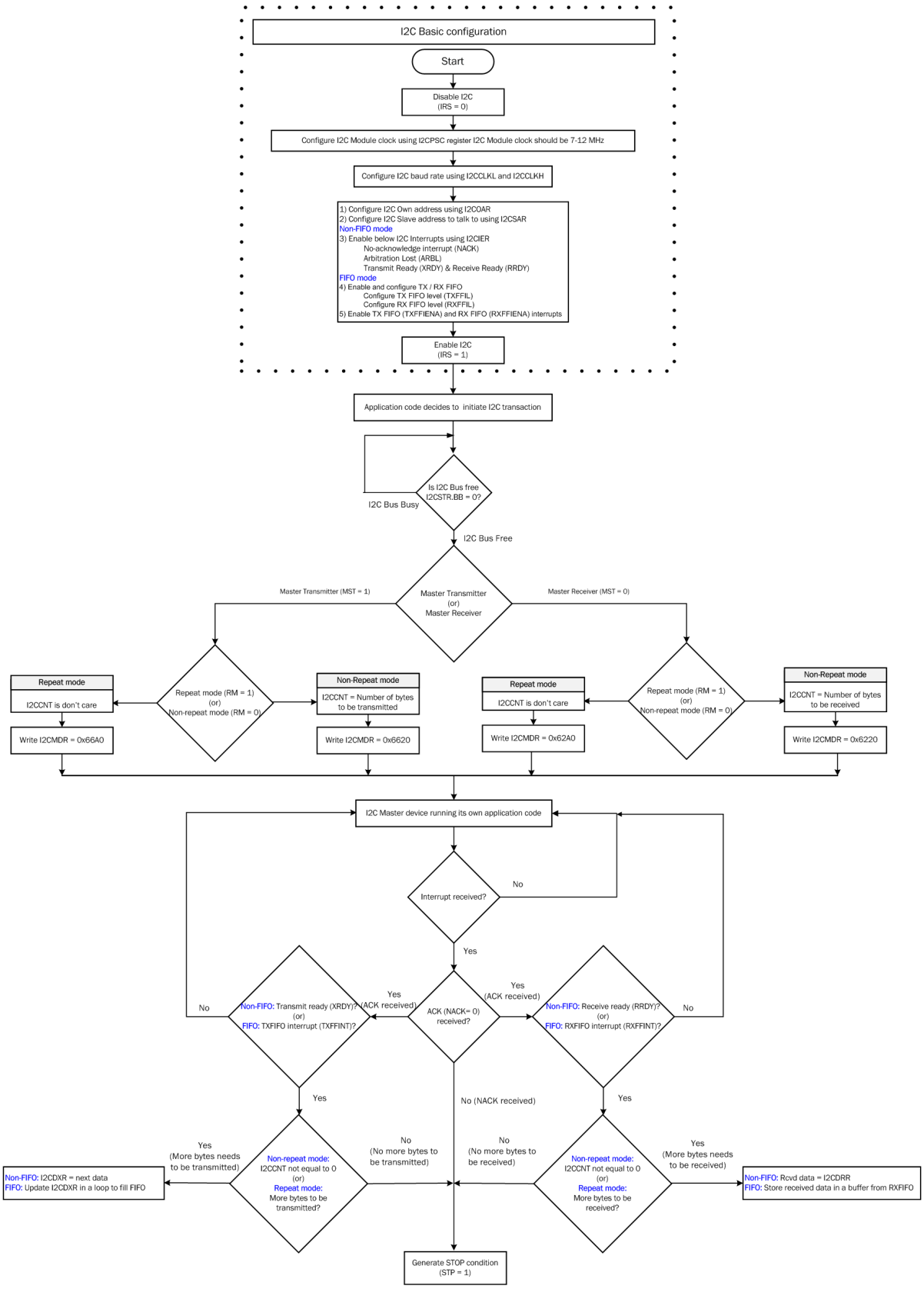



**Figure 20-7. I2C Master TX / RX Flowchart**



Copyright © 2024 Texas Instruments Incorporated

### 20.3.5 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 20-8:

- 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.

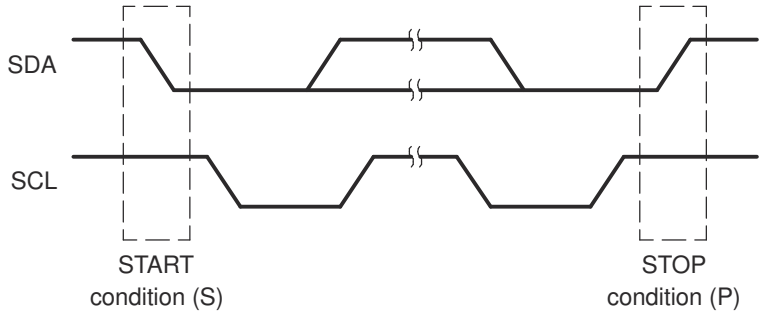



**Figure 20-8. I2C Module START and STOP Conditions**

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 20.7.

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.



Copyright © 2024 Texas Instruments Incorporated

### 20.3.6 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.

### 20.3.7 Serial Data Formats

Figure 20-9 shows an example of a data transfer on the I2C bus. The I2C module supports 1 to 8-bit data values. In Figure 20-9, 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 20-9 is the 7-bit addressing format. The I2C module supports the formats shown in Figure 20-10 through Figure 20-12 and described in the paragraphs that follow the figures.

**Note**

In Figure 20-9 through Figure 20-12, n = the number of data bits (from 1 to 8) specified by the bit count (BC) field of I2CMDR.

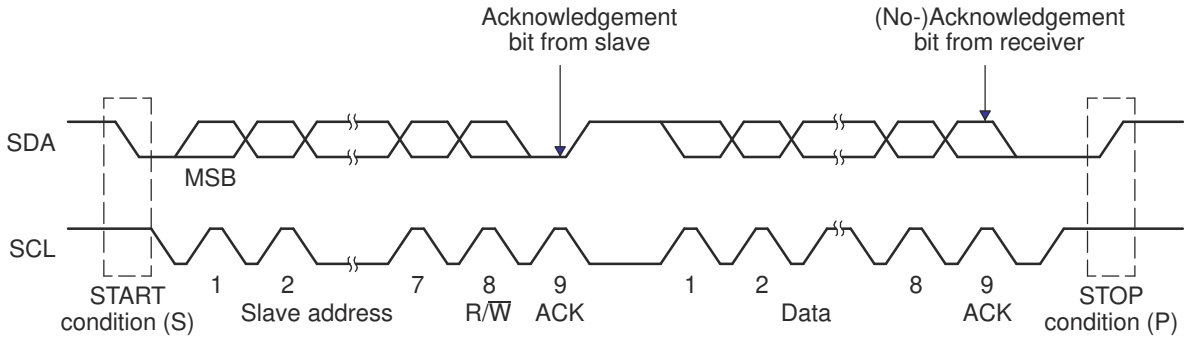



**Figure 20-9. I2C Module Data Transfer (7-Bit Addressing with 8-bit Data Configuration Shown)**



Copyright © 2024 Texas Instruments Incorporated

#### 20.3.7.1 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 20-10), the first byte after a START condition (S) consists of a 7-bit slave address followed by a R/$\overline{W}$ bit. R/$\overline{W}$ determines the direction of the data:

- R/$\overline{W}$ = 0: The I2C master writes (transmits) data to the addressed slave. This can be achieved by setting I2CMDR.TRX = 1 (Transmitter mode)
- R/$\overline{W}$ = 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/$\overline{W}$ 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.

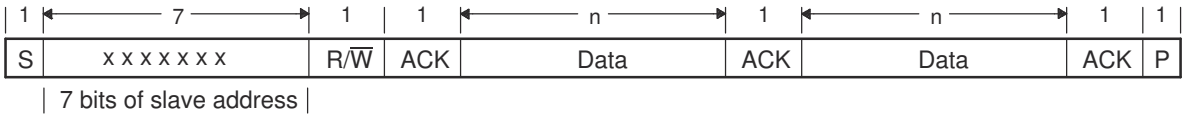



**Figure 20-10. I2C Module 7-Bit Addressing Format (FDF = 0, XA = 0 in I2CMDR)**

#### 20.3.7.2 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 20-11) 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/$\overline{W}$. 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.

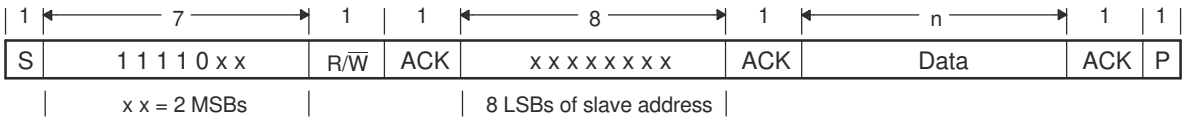



**Figure 20-11. I2C Module 10-Bit Addressing Format (FDF = 0, XA = 1 in I2CMDR)**



Copyright © 2024 Texas Instruments Incorporated

#### 20.3.7.3 Free Data Format

The free data format can be enabled by setting I2CMDR. FDF = 1.

In this format (see Figure 20-12), 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.

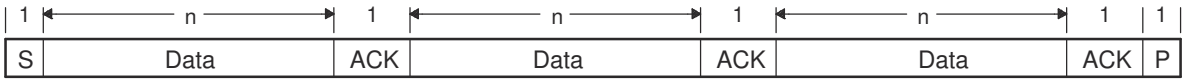



**Figure 20-12. I2C Module Free Data Format (FDF = 1 in I2CMDR)**

**Note**

The free data format is not supported in the digital loopback mode (I2CMDR.DLB = 1).

**Table 20-4. 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:<br>TRX = 1: The I2C module is a transmitter.<br>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.<br>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.<br>TRX = 1: The I2C module is a transmitter. |

#### 20.3.7.4 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 20-13 shows a repeated START condition in the 7-bit addressing format.

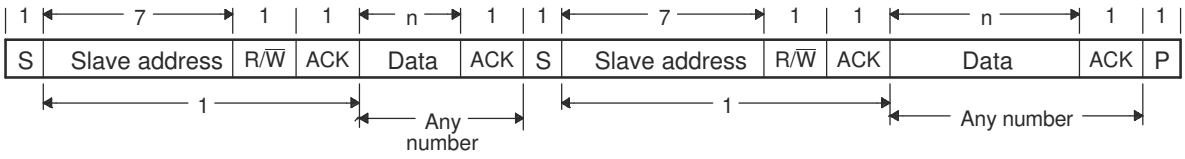



**Figure 20-13. Repeated START Condition (in This Case, 7-Bit Addressing Format)**

**Note**

In Figure 20-13, n = the number of data bits (from 1 to 8) specified by the bit count (BC) field of I2CMDR.



Copyright © 2024 Texas Instruments Incorporated

### 20.3.8 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 20-14 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.

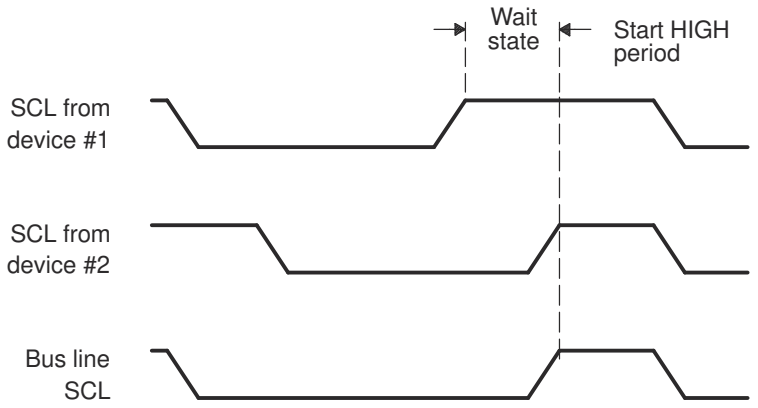



**Figure 20-14. Synchronization of Two I2C Clock Generators During Arbitration**



Copyright © 2024 Texas Instruments Incorporated

### 20.3.9 Arbitration

If two or more master-transmitters attempt to start a transmission on the same bus at approximately the same time, an arbitration procedure is invoked. The arbitration procedure uses the data presented on the serial data bus (SDA) by the competing transmitters. Figure 20-15 illustrates the arbitration procedure between two devices. The first master-transmitter that releases the SDA line high is overruled by another master-transmitter that drives the SDA low. The arbitration procedure gives priority to the device that transmits the serial data stream with the lowest binary value. If two or more devices send identical first bytes, arbitration continues on the subsequent bytes.

If the I2C module is the losing master, the I2c module switches to the slave-receiver mode, sets the arbitration lost (ARBL) flag, and generates the arbitration-lost interrupt request.

If during a serial transfer the arbitration procedure is still in progress when a repeated START condition or a STOP condition is transmitted to SDA, the master-transmitters involved must send the repeated START condition or the STOP condition at the same position in the format frame. Arbitration is not allowed between:

- A repeated START condition and a data bit
- A STOP condition and a data bit
- A repeated START condition and a STOP condition

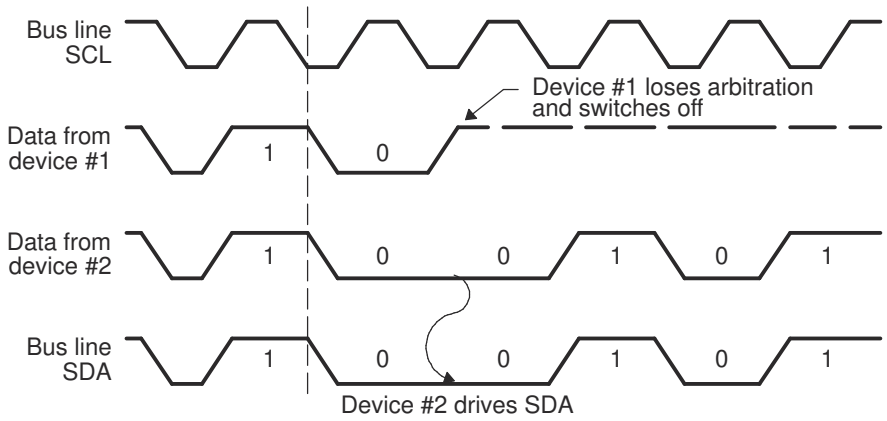



**Figure 20-15. Arbitration Procedure Between Two Master-Transmitters**



Copyright © 2024 Texas Instruments Incorporated

### 20.3.10 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 * (SYSCLK) / (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 20-16 shows the signal routing in digital loopback mode.

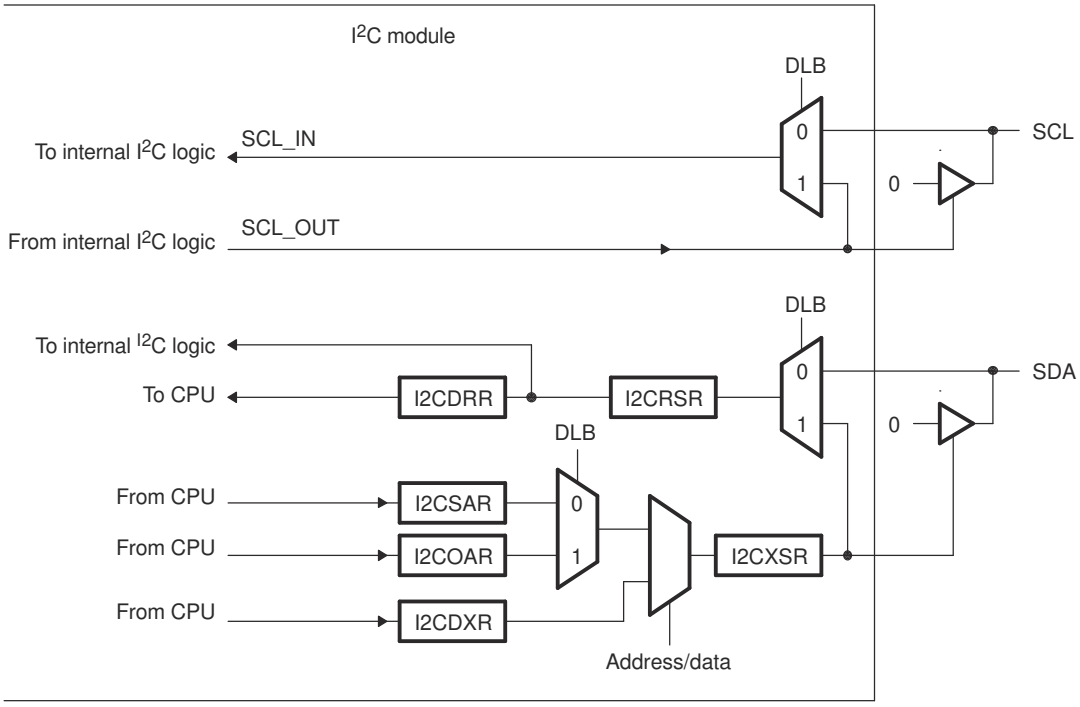



**Figure 20-16. Pin Diagram Showing the Effects of the Digital Loopback Mode (DLB) Bit**

**Note**

The free data format (I2CMDR.FDF = 1) is not supported in digital loopback mode.



Copyright © 2024 Texas Instruments Incorporated

### 20.3.11 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 20-5 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 I2CMDR is cleared
2. SCL is held low
3. The NACK in I2CSTR is set

**Table 20-5. Ways to Generate a NACK Bit**

| I2C Module Condition | NACK Bit Generation Options |
|---|---|
| Slave-receiver modes | Allow an overrun condition (RSFULL = 1 in I2CSTR)<br>Reset the module (IRS = 0 in I2CMDR)<br>Set the NACKMOD bit of I2CMDR before the rising edge of the last data bit you intend to receive |
| Master-receiver mode AND<br>Repeat mode (RM = 1 in I2CMDR) | Generate a STOP condition (STP = 1 in I2CMDR)<br>Reset the module (IRS = 0 in I2CMDR)<br>Set the NACKMOD bit of I2CMDR before the rising edge of the last data bit you intend to receive |
| Master-receiver mode AND<br>Nonrepeat mode<br>(RM = 0 in I2CMDR) | If STP = 1 in I2CMDR, allow the internal data counter to count down to 0 and thus force a STOP condition<br>If STP = 0, make STP = 1 to generate a STOP condition<br>Reset the module (IRS = 0 in I2CMDR)<br>Set STP = 1 to generate a STOP condition<br>Set the NACKMOD bit of I2CMDR before the rising edge of the last data bit you intend to receive |

## 20.4 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 20.4.1.
2. I2C FIFO interrupt: Possible I2C FIFO interrupt sources that can trigger this interrupt are described in Section 20.4.2



Copyright © 2024 Texas Instruments Incorporated

### 20.4.1 Basic I2C Interrupt Requests

The I2C module generates the interrupt requests described in Table 20-6. As shown in Figure 20-17, 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 20-6. Descriptions of the Basic I2C Interrupt Requests**

| I2C Interrupt Request | Interrupt Source |
|---|---|
| XRDYINT | Transmit ready condition: The data transmit register (I2CDXR) is ready to accept new data because the previous data has been copied from I2CDXR to the transmit shift register (I2CXSR).<br>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. |
| RRDYINT | Receive ready condition: The data receive register (I2CDRR) is ready to be read because data has been copied from the receive shift register (I2CRSR) to I2CDRR.<br>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. |
| ARDYINT | Register-access ready condition: The I2C module registers are ready to be accessed because the previously programmed address, data, and command values have been used.<br>The specific events that generate ARDYINT are the same events that set the ARDY bit of I2CSTR.<br>As an alternative to using ARDYINT, the CPU can poll the ARDY bit. |
| NACKINT | No-acknowledgment condition: The I2C module is configured as a master-transmitter and did not received acknowledgment from the slave-receiver.<br>As an alternative to using NACKINT, the CPU can poll the NACK bit of I2CSTR. |
| ARBLINT | Arbitration-lost condition: The I2C module has lost an arbitration contest with another master-transmitter.<br>As an alternative to using ARBLINT, the CPU can poll the ARBL bit of I2CSTR. |
| SCDINT | Stop condition detected: A STOP condition was detected on the I2C bus.<br>As an alternative to using SCDINT, the CPU can poll the SCD bit of the status register, I2CSTR. |
| AASINT | Addressed as slave condition: The I2C has been addressed as a slave device by another master on the I2C bus.<br>As an alternative to using AASINT, the CPU can poll the AAS bit of the status register, I2CSTR. |



Copyright © 2024 Texas Instruments Incorporated

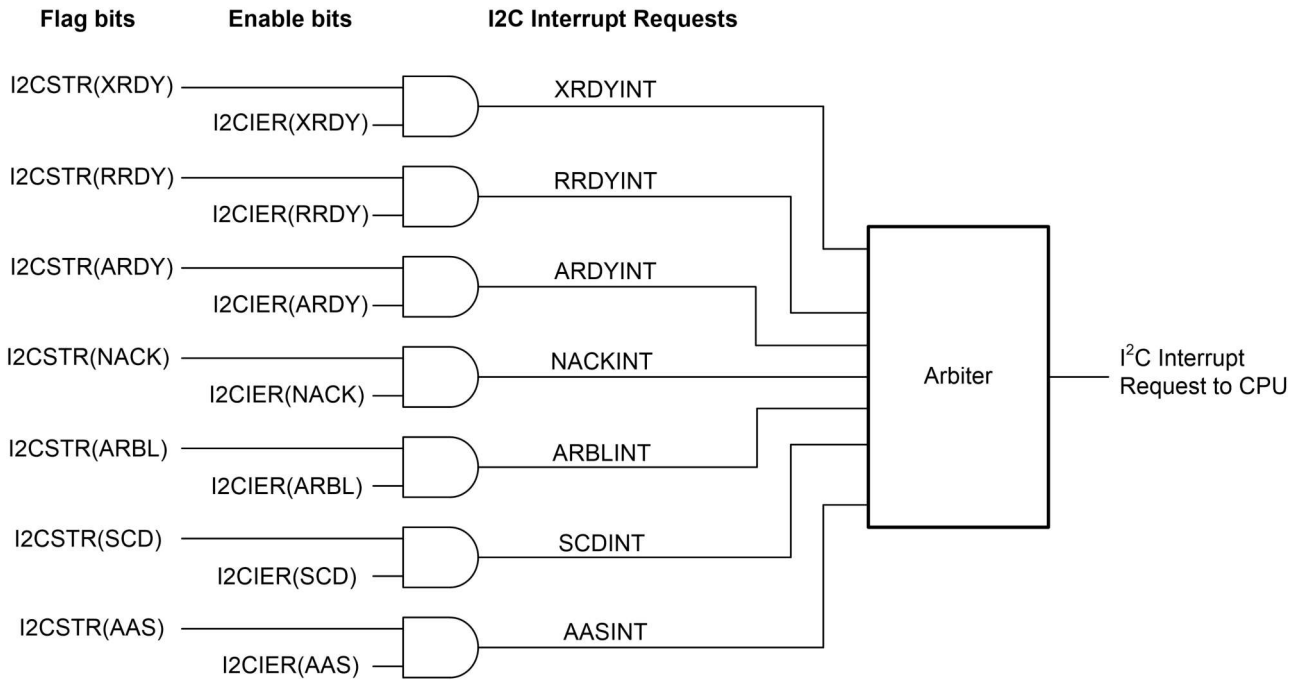



**Figure 20-17. Enable Paths of the I2C Interrupt Requests**

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 I2C module has a backwards compatibility bit (BC) in the I2CEMDR register. The timing diagram in Figure 20-18 demonstrates the effect the backwards compatibility bit has on I2C module registers and interrupts when configured as a slave-transmitter.



Copyright © 2024 Texas Instruments Incorporated

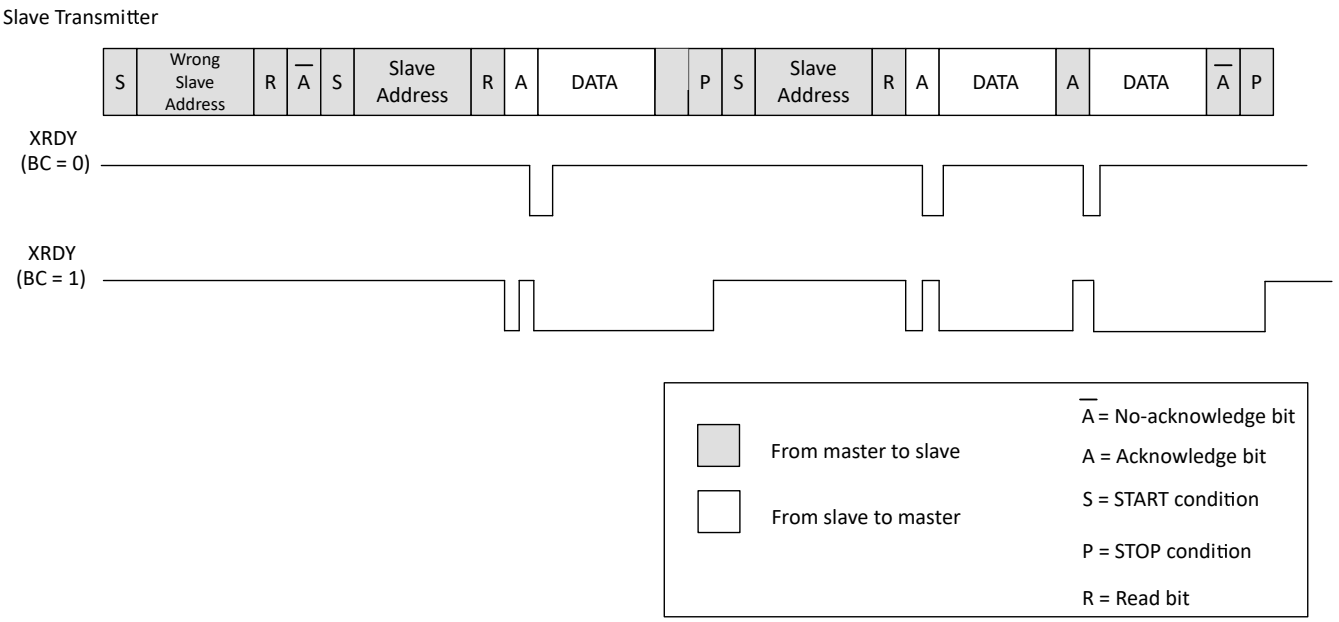



**Figure 20-18. Backwards Compatibility Mode and Forward Compatibility Bit, Slave Transmitter**



Copyright © 2024 Texas Instruments Incorporated

### 20.4.2 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 20-19 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.

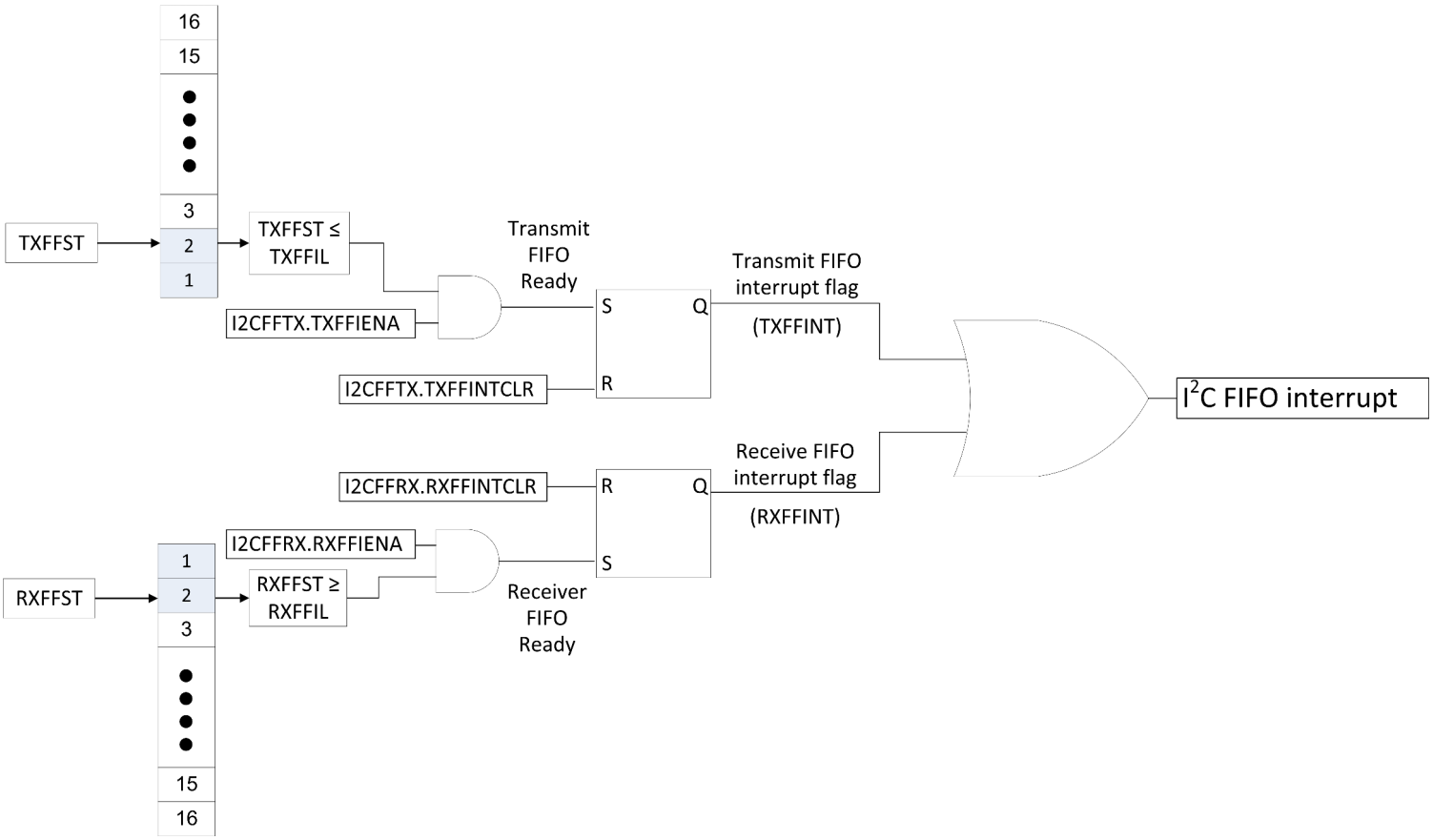



**Figure 20-19. I2C FIFO Interrupt**

## 20.5 Resetting or Disabling the I2C Module

You can reset or disable the I2C module in two ways:

- Write 0 to the I2C reset bit (IRS) in the I2C mode register (I2CMDR). All status bits (in I2CSTR) are forced to 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.



Copyright © 2024 Texas Instruments Incorporated

## 20.6 Software

### *20.6.1 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 C2000Ware Examples.

#### 20.6.1.1 C28x-I2C Library source file for FIFO interrupts

FILE: i2cLib_FIFO_controller_interrupt.c

#### 20.6.1.2 C28x-I2C Library source file for FIFO using polling

FILE: i2cLib_FIFO_polling.c

#### 20.6.1.3 C28x-I2C Library source file for FIFO interrupts

FILE: i2cLib_FIFO_controller_target_interrupt.c

#### 20.6.1.4 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.

This example project has support for migration across our C2000 device families. If you are wanting to build this project from launchpad or controlCARD, please specify in the .syscfg file the board you're using. At any time you can select another device to migrate this example. *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

#### 20.6.1.5 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



Copyright © 2024 Texas Instruments Incorporated

- *i2cMsgIn* - Message containing data read from EEPROM

#### 20.6.1.6 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

#### 20.6.1.7 I2C EEPROM

FILE: i2c_ex4_eeprom_polling.c

This program will shows 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.

#### 20.6.1.8 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 using 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

\b External \b Connections on launchpad should be made as shown below \n 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*



Copyright © 2024 Texas Instruments Incorporated

- *I2CA_RXdata*
- *I2CB_TXdata*
- *I2CB_RXdata* stream for error checking

#### 20.6.1.9 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_SCLA | SCL SDA | DEVICE_GPIO_PIN_SDAA | 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

## 20.7 I2C Registers

This section describes the I2C Module Registers.

### *20.7.1 I2C Base Addresses*

**Table 20-7. I2C Base Address Table (C28)**

| Bit Field Name | | DriverLib Name | Base Address | CPU1 | CPU2 | DMA | CLA | Pipeline Protected |
|---|---|---|---|---|---|---|---|---|
| Instance | Structure | | | | | | | |
| I2caRegs | I2C_REGS | I2CA_BASE | 0x0000_7300 | YES | YES | - | - | YES |
| I2cbRegs | I2C_REGS | I2CB_BASE | 0x0000_7340 | YES | YES | - | - | YES |



Copyright © 2024 Texas Instruments Incorporated

### 20.7.2 I2C_REGS Registers

Table 20-8 lists the memory-mapped registers for the I2C_REGS registers. All register offset addresses not listed in Table 20-8 should be considered as reserved locations and the register contents should not be modified.

**Table 20-8. I2C_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | I2COAR | I2C Own address | | Go |
| 1h | I2CIER | I2C Interrupt Enable | | Go |
| 2h | I2CSTR | I2C Status | | Go |
| 3h | I2CCLKL | I2C Clock low-time divider | | Go |
| 4h | I2CCLKH | I2C Clock high-time divider | | Go |
| 5h | I2CCNT | I2C Data count | | Go |
| 6h | I2CDRR | I2C Data receive | | Go |
| 7h | I2CSAR | I2C Slave address | | Go |
| 8h | I2CDXR | I2C Data Transmit | | Go |
| 9h | I2CMDR | I2C Mode | | Go |
| Ah | I2CISRC | I2C Interrupt Source | | Go |
| Bh | I2CEMDR | I2C Extended Mode | | Go |
| Ch | I2CPSC | I2C Prescaler | | Go |
| 20h | I2CFFTX | I2C FIFO Transmit | | Go |
| 21h | I2CFFRX | I2C FIFO Receive | | Go |

Complex bit access types are encoded to fit into small table cells. Table 20-9 shows the codes that are used for access types in this section.

**Table 20-9. I2C_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| R-0 | R<br>-0 | Read<br>Returns 0s |
| Write Type | | |
| W | W | Write |
| W1C | W<br>1C | Write<br>1 to clear |
| W1S | W<br>1S | Write<br>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. |



Copyright © 2024 Texas Instruments Incorporated

### 20.7.2.1 I2COAR Register (Offset = 0h) [Reset = 0000h]

I2COAR is shown in Figure 20-20 and described in Table 20-10.

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 20-20. 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 20-10. 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):<br>00h-7Fh Bits 6-0 provide the 7-bit slave address of the I2C module. Write 0s to bits 9-7.<br>In 10-bit addressing mode (XA = 1 in I2CMDR):<br>000h-3FFh Bits 9-0 provide the 10-bit slave address of the I2C module.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 20.7.2.2 I2CIER Register (Offset = 1h) [Reset = 0000h]

I2CIER is shown in Figure 20-21 and described in Table 20-11.

Return to the Summary Table.

I2CIER is used by the CPU to individually enable or disable I2C interrupt requests.

**Figure 20-21. 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 | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 20-11. 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<br>Reset type: SYSRSn<br>0h (R/W) = Interrupt request disabled<br>1h (R/W) = Interrupt request enabled |
| 5 | SCD | R/W | 0h | Stop condition detected interrupt enable<br>Reset type: SYSRSn<br>0h (R/W) = Interrupt request disabled<br>1h (R/W) = Interrupt request enabled |
| 4 | XRDY | R/W | 0h | Transmit-data-ready interrupt enable bit.<br>This bit should not be set when using FIFO mode.<br>Reset type: SYSRSn<br>0h (R/W) = Interrupt request disabled<br>1h (R/W) = Interrupt request enabled |
| 3 | RRDY | R/W | 0h | Receive-data-ready interrupt enable bit.<br>This bit should not be set when using FIFO mode.<br>Reset type: SYSRSn<br>0h (R/W) = Interrupt request disabled<br>1h (R/W) = Interrupt request enabled |
| 2 | ARDY | R/W | 0h | Register-access-ready interrupt enable<br>Reset type: SYSRSn<br>0h (R/W) = Interrupt request disabled<br>1h (R/W) = Interrupt request enabled |
| 1 | NACK | R/W | 0h | No-acknowledgment interrupt enable<br>Reset type: SYSRSn<br>0h (R/W) = Interrupt request disabled<br>1h (R/W) = Interrupt request enabled |
| 0 | ARBL | R/W | 0h | Arbitration-lost interrupt enable<br>Reset type: SYSRSn<br>0h (R/W) = Interrupt request disabled<br>1h (R/W) = Interrupt request enabled |



Copyright © 2024 Texas Instruments Incorporated

### 20.7.2.3 I2CSTR Register (Offset = 2h) [Reset = 0410h]

I2CSTR is shown in Figure 20-22 and described in Table 20-12.

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 20-22. 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 | | SCD | XRDY | RRDY | ARDY | NACK | ARBL |
| R-0h | | R/W1C-0h | R-1h | R/W1C-0h | R/W1C-0h | R/W1C-0h | R/W1C-0h |

**Table 20-12. I2CSTR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RESERVED | R | 0h | Reserved |
| 14 | SDIR | R/W1C | 0h | Slave direction bit<br>Reset type: SYSRSn<br>0h (R/W) = I2C is not addressed as a slave transmitter. SDIR is cleared by one of the following events:<br>- It is manually cleared. To clear this bit, write a 1 to it.<br>- Digital loopback mode is enabled.<br>- A START or STOP condition occurs on the I2C bus.<br>1h (R/W) = I2C is addressed as a slave transmitter. |
| 13 | NACKSNT | R/W1C | 0h | NACK sent bit.<br>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<br>Reset type: SYSRSn<br>0h (R/W) = NACK not sent. NACKSNT bit is cleared by any one of the following events:<br>- It is manually cleared. To clear this bit, write a 1 to it.<br>- The I2C module is reset (either when 0 is written to the IRS bit of I2CMDR or when the whole device is reset).<br>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.<br>BB indicates whether the I2C-bus is busy or is free for another data transfer. See the paragraph following the table for more information<br>Reset type: SYSRSn<br>0h (R/W) = Bus free. BB is cleared by any one of the following events:<br>- The I2C module receives or transmits a STOP bit (bus free).<br>- The I2C module is reset.<br>1h (R/W) = Bus busy: The I2C module has received or transmitted a START bit on the bus. |



Copyright © 2024 Texas Instruments Incorporated

**Table 20-12. I2CSTR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 11 | RSFULL | R | 0h | Receive shift register full bit.<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = No overrun detected. RSFULL is cleared by any one of the following events:<br>- I2CDRR is read by the CPU. Emulator reads of the I2CDRR do not affect this bit.<br>- The I2C module is reset.<br>1h (R/W) = Overrun detected |
| 10 | XSMT | R | 1h | Transmit shift register empty bit.<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = Underflow detected (empty)<br>1h (R/W) = No underflow detected (not empty). XSMT is set by one of the following events:<br>- Data is written to I2CDXR.<br>- The I2C module is reset |
| 9 | AAS | R | 0h | Addressed-as-slave bit<br>Reset type: SYSRSn<br>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.<br>1h (R/W) = The I2C module has recognized its own slave address or an address of all zeros (general call). |
| 8 | AD0 | R | 0h | Address 0 bits<br>Reset type: SYSRSn<br>0h (R/W) = AD0 has been cleared by a START or STOP condition.<br>1h (R/W) = An address of all zeros (general call) is detected. |
| 7-6 | RESERVED | R | 0h | Reserved |
| 5 | SCD | R/W1C | 0h | Stop condition detected bit.<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = STOP condition not detected since SCD was last cleared. SCD is cleared by any one of the following events:<br>- 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.<br>- SCD is manually cleared. To clear this bit, write a 1 to it.<br>- The I2C module is reset.<br>1h (R/W) = A STOP condition has been detected on the I2C bus. |



Copyright © 2024 Texas Instruments Incorporated

**Table 20-12. I2CSTR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 4 | XRDY | R | 1h | Transmit-data-ready interrupt flag bit.<br>When not in FIFO mode, XRDY indicates that the data transmit register (I2CDXR) is ready to accept new data because the previous data has been copied from I2CDXR to the transmit shift register (I2CXSR). The CPU can poll XRDY or use the XRDY interrupt request When in FIFO mode, use TXFFINT instead.<br>Reset type: SYSRSn<br>0h (R/W) = I2CDXR not ready. XRDY is cleared when data is written to I2CDXR.<br>1h (R/W) = I2CDXR ready: Data has been copied from I2CDXR to I2CXSR.<br>XRDY is also forced to 1 when the I2C module is reset. |
| 3 | RRDY | R/W1C | 0h | Receive-data-ready interrupt flag bit.<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = I2CDRR not ready. RRDY is cleared by any one of the following events:<br>- I2CDRR is read by the CPU. Emulator reads of the I2CDRR do not affect this bit.<br>- RRDY is manually cleared. To clear this bit, write a 1 to it.<br>- The I2C module is reset.<br>1h (R/W) = I2CDRR ready: Data has been copied from I2CRSR to I2CDRR. |
| 2 | ARDY | R/W1C | 0h | Register-access-ready interrupt flag bit (only applicable when the I2C module is in the master mode).<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = The registers are not ready to be accessed. ARDY is cleared by any one of the following events:<br>- The I2C module starts using the current register contents.<br>- ARDY is manually cleared. To clear this bit, write a 1 to it.<br>- The I2C module is reset.<br>1h (R/W) = The registers are ready to be accessed.<br>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).<br>In the repeat mode (RM = 1): ARDY is set at the end of each byte transmitted from I2CDXR. |



Copyright © 2024 Texas Instruments Incorporated

**Table 20-12. I2CSTR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1 | NACK | R/W1C | 0h | No-acknowledgment interrupt flag bit.<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = ACK received/NACK not received. This bit is cleared by any one of the following events:<br>- An acknowledge bit (ACK) has been sent by the slave receiver.<br>- NACK is manually cleared. To clear this bit, write a 1 to it.<br>- 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.<br>- The I2C module is reset.<br>1h (R/W) = NACK bit received. The hardware detects that a no-acknowledge (NACK) bit has been received.<br>Note: While the I2C module performs a general call transfer, NACK is 1, even if one or more slaves send acknowledgment. |
| 0 | ARBL | R/W1C | 0h | Arbitration-lost interrupt flag bit (only applicable when the I2C module is a master-transmitter).<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = Arbitration not lost. AL is cleared by any one of the following events:<br>- AL is manually cleared. To clear this bit, write a 1 to it.<br>- The CPU reads the interrupt source register (I2CISRC) and the register contains the code for an<br>AL interrupt. Emulator reads of the I2CISRC do not affect this bit.<br>- The I2C module is reset.<br>1h (R/W) = Arbitration lost. AL is set by any one of the following events:<br>- The I2C module senses that it has lost an arbitration with two or more competing transmitters that started a transmission almost simultaneously.<br>- The I2C module attempts to start a transfer while the BB (bus busy) bit is set to 1.<br>When AL becomes 1, the MST and STP bits of I2CMDR are cleared, and the I2C module becomes a slave-receiver. |



Copyright © 2024 Texas Instruments Incorporated

### 20.7.2.4 I2CCLKL Register (Offset = 3h) [Reset = 0000h]

I2CCLKL is shown in Figure 20-23 and described in Table 20-13.

Return to the Summary Table.

I2C Clock low-time divider

**Figure 20-23. 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 20-13. I2CCLKL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | I2CCLKL | R/W | 0h | Clock low-time divide-down value.<br>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.<br>Note: These bits must be set to a non-zero value for proper I2C clock generation.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 20.7.2.5 I2CCLKH Register (Offset = 4h) [Reset = 0000h]

I2CCLKH is shown in Figure 20-24 and described in Table 20-14.

Return to the Summary Table.

I2C Clock high-time divider

**Figure 20-24. 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 20-14. I2CCLKH Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | I2CCLKH | R/W | 0h | Clock high-time divide-down value.<br>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.<br>Note: These bits must be set to a non-zero value for proper I2C clock generation.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 20.7.2.6 I2CCNT Register (Offset = 5h) [Reset = 0000h]

I2CCNT is shown in Figure 20-25 and described in Table 20-15.

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 20-25. 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 20-15. I2CCNT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | I2CCNT | R/W | 0h | Data count value. I2CCNT indicates the number of data bytes to transfer or receive.<br>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.<br>The value in I2CCNT is a don't care when the RM bit in I2CMDR is set to 1.<br>Reset type: SYSRSn<br>0h (R/W) = data count value is 65536<br>1h (R/W) = data count value is 1<br>2h (R/W) = data count value is 2<br>FFFFh (R/W) = data count value is 65535 |



Copyright © 2024 Texas Instruments Incorporated

### 20.7.2.7 I2CDRR Register (Offset = 6h) [Reset = 0000h]

I2CDRR is shown in Figure 20-26 and described in Table 20-16.

Return to the Summary Table.

I2CDRR is a 16-bit register used by the CPU to read received data. The I2C module can receive a data byte with 1 to 8 bits. The number of bits is selected with the bit count (BC) bits in I2CMDR. One bit at a time is shifted in from the SDA pin to the receive shift register (I2CRSR). When a complete data byte has been received, the I2C module copies the data byte from I2CRSR to I2CDRR. The CPU cannot access I2CRSR directly.
If a data byte with fewer than 8 bits is in I2CDRR, the data value is right-justified, and the other bits of I2CDRR(7-0) are undefined. For example, if BC = 011 (3-bit data size), the receive data is in I2CDRR(2-0), and the content of I2CDRR(7-3) is undefined.
When in the receive FIFO mode, the I2CDRR register acts as the receive FIFO buffer.

**Figure 20-26. 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 20-16. I2CDRR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R | 0h | Reserved |
| 7-0 | DATA | R | 0h | Receive data<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 20.7.2.8 I2CSAR Register (Offset = 7h) [Reset = 03FFh]

I2CSAR is shown in Figure 20-27 and described in Table 20-17.

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 20-27. 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 20-17. 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):<br>00h-7Fh Bits 6-0 provide the 7-bit slave address that the I2C module transmits when it is in the master-transmitter<br>mode. Write 0s to bits 9-7.<br>In 10-bit addressing mode (XA = 1 in I2CMDR):<br>000h-3FFh Bits 9-0 provide the 10-bit slave address that the I2C module transmits when it is in the master transmitter mode.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 20.7.2.9 I2CDXR Register (Offset = 8h) [Reset = 0000h]

I2CDXR is shown in Figure 20-28 and described in Table 20-18.

Return to the Summary Table.

The CPU writes transmit data to I2CDXR. This 16-bit register accepts a data byte with 1 to 8 bits. Before writing to I2CDXR, specify how many bits are in a data byte by loading the appropriate value into the bit count (BC) bits of I2CMDR. When writing a data byte with fewer than 8 bits, make sure the value is right-aligned in I2CDXR. After a data byte is written to I2CDXR, the I2C module copies the data byte to the transmit shift register (I2CXSR). The CPU cannot access I2CXSR directly. From I2CXSR, the I2C module shifts the data byte out on the SDA pin, one bit at a time.
When in the transmit FIFO mode, the I2CDXR register acts as the transmit FIFO buffer.

**Figure 20-28. 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 20-18. I2CDXR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R | 0h | Reserved |
| 7-0 | DATA | R/W | 0h | Transmit data<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 20.7.2.10 I2CMDR Register (Offset = 9h) [Reset = 0000h]

I2CMDR is shown in Figure 20-29 and described in Table 20-19.

Return to the Summary Table.

The I2C mode register (I2CMDR) is a 16-bit register that contains the control bits of the I2C module.

**Figure 20-29. I2CMDR Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| NACKMOD | FREE | STT | RESERVED | STP | MST | TRX | XA |
| R/W-0h | R/W-0h | R/W-0h | R-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RM | DLB | IRS | STB | FDF | BC | | |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | | |

**Table 20-19. I2CMDR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | NACKMOD | R/W | 0h | NACK mode bit. This bit is only applicable when the I2C module is acting as a receiver.<br>Reset type: SYSRSn<br>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.<br>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<br>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.<br>Important: To send a NACK bit in the next acknowledge cycle, you must set NACKMOD before the rising edge of the last data bit. |
| 14 | FREE | R/W | 0h | This bit controls the action taken by the I2C module when a debugger breakpoint is encountered.<br>Reset type: SYSRSn<br>0h (R/W) = When I2C module is master:<br>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.<br>When I2C module is slave:<br>A breakpoint forces the I2C module to stop when the current transmission/reception is complete.<br>1h (R/W) = The I2C module runs free<br>that is, it continues to operate when a breakpoint occurs. |
| 13 | STT | R/W | 0h | 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.<br>Reset type: SYSRSn<br>0h (R/W) = In the master mode, STT is automatically cleared after the START condition has been generated.<br>1h (R/W) = In the master mode, setting STT to 1 causes the I2C module to generate a START condition on the I2C-bus |
| 12 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 20-19. I2CMDR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 11 | STP | R/W | 0h | STOP condition bit (only applicable when the I2C module is a master).<br>In the master mode, the RM,STT, and STP bits determine when the I2C module starts and stops data transmissions.<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = STP is automatically cleared after the STOP condition has been generated<br>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. |
| 10 | MST | R/W | 0h | Master mode bit.<br>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<br>Reset type: SYSRSn<br>0h (R/W) = Slave mode. The I2C module is a slave and receives the serial clock from the master.<br>1h (R/W) = Master mode. The I2C module is a master and generates the serial clock on the SCL pin. |
| 9 | TRX | R/W | 0h | Transmitter mode bit.<br>When relevant, TRX selects whether the I2C module is in the transmitter mode or the receiver mode.<br>Reset type: SYSRSn<br>0h (R/W) = Receiver mode. The I2C module is a receiver and receives data on the SDA pin.<br>1h (R/W) = Transmitter mode. The I2C module is a transmitter and transmits data on the SDA pin. |
| 8 | XA | R/W | 0h | Expanded address enable bit.<br>Reset type: SYSRSn<br>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).<br>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). |
| 7 | RM | R/W | 0h | Repeat mode bit (only applicable when the I2C module is a master-transmitter or master-receiver).<br>The RM, STT, and STP bits determine when the I2C module starts and stops data transmissions<br>Reset type: SYSRSn<br>0h (R/W) = Nonrepeat mode. The value in the data count register (I2CCNT) determines how many bytes are received/transmitted by the I2C module.<br>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. |



Copyright © 2024 Texas Instruments Incorporated

**Table 20-19. I2CMDR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 6 | DLB | R/W | 0h | Digital loopback mode bit.<br>Reset type: SYSRSn<br>0h (R/W) = Digital loopback mode is disabled.<br>1h (R/W) = Digital loopback mode is enabled. For proper operation in this mode, the MST bit must be 1.<br>In the digital loopback mode, data transmitted out of I2CDXR is received in I2CDRR after n device cycles by an internal path, where: n = ((I2C input clock frequency/module clock frequency) x 8)<br>The transmit clock is also the receive clock. The address transmitted on the SDA pin is the address in I2COAR.<br>Note: The free data format (FDF = 1) is not supported in the digital loopback mode. |
| 5 | IRS | R/W | 0h | I2C module reset bit.<br>Reset type: SYSRSn<br>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.<br>1h (R/W) = The I2C module is enabled. This has the effect of releasing the I2C bus if the I2C peripheral is holding it. |
| 4 | STB | R/W | 0h | 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.<br>Reset type: SYSRSn<br>0h (R/W) = The I2C module is not in the START byte mode.<br>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:<br>1. A START condition<br>2. A START byte (0000 0001b)<br>3. A dummy acknowledge clock pulse<br>4. A repeated START condition<br>Then, as normal, the I2C module sends the slave address that is in I2CSAR. |
| 3 | FDF | R/W | 0h | Free data format mode bit.<br>Reset type: SYSRSn<br>0h (R/W) = Free data format mode is disabled. Transfers use the 7-/10-bit addressing format selected by the XA bit.<br>1h (R/W) = Free data format mode is enabled. Transfers have the free data (no address) format described in Section 9.2.5.<br>The free data format is not supported in the digital loopback mode (DLB=1). |



Copyright © 2024 Texas Instruments Incorporated

**Table 20-19. I2CMDR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 2-0 | BC | R/W | 0h | Bit count bits.<br>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.<br>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<br>Reset type: SYSRSn<br>0h (R/W) = 8 bits per data byte<br>1h (R/W) = 1 bit per data byte<br>2h (R/W) = 2 bits per data byte<br>3h (R/W) = 3 bits per data byte<br>4h (R/W) = 4 bits per data byte<br>5h (R/W) = 5 bits per data byte<br>6h (R/W) = 6 bits per data byte<br>7h (R/W) = 7 bits per data byte |



Copyright © 2024 Texas Instruments Incorporated

### 20.7.2.11 I2CISRC Register (Offset = Ah) [Reset = 0000h]

I2CISRC is shown in Figure 20-30 and described in Table 20-20.

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 20-30. 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 20-20. 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<br>These reserved bit locations should always be written as zeros.<br>Reset type: SYSRSn |
| 7-3 | RESERVED | R | 0h | Reserved |
| 2-0 | INTCODE | R | 0h | Interrupt code bits.<br>The binary code in INTCODE indicates the event that generated an I2C interrupt.<br>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.<br>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.<br>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.<br>Emulator reads will not affect the state of this field or of the status bits in the I2CSTR register.<br>Reset type: SYSRSn<br>0h (R/W) = None<br>1h (R/W) = Arbitration lost<br>2h (R/W) = No-acknowledgment condition detected<br>3h (R/W) = Registers ready to be accessed<br>4h (R/W) = Receive data ready<br>5h (R/W) = Transmit data ready<br>6h (R/W) = Stop condition detected<br>7h (R/W) = Addressed as slave |



Copyright © 2024 Texas Instruments Incorporated

### 20.7.2.12 I2CEMDR Register (Offset = Bh) [Reset = 0001h]

I2CEMDR is shown in Figure 20-31 and described in Table 20-21.

Return to the Summary Table.

I2C Extended Mode

**Figure 20-31. I2CEMDR Register**

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

**Table 20-21. I2CEMDR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-1 | RESERVED | R | 0h | Reserved |
| 0 | BC | R/W | 1h | Backwards compatibility mode.<br>This bit affects the timing of the transmit status bits (XRDY and XSMT) in the I2CSTR register when in slave transmitter mode. Check Backwards Compatibility Mode Bit, Slave Transmitter diagram for more details.<br>Reset type: SYSRSn<br>0h (R/W) = See the 'Backwards Compatibility Mode Bit, Slave Transmitter' Figure for details.<br>1h (R/W) = See the 'Backwards Compatibility Mode Bit, Slave Transmitter' Figure for details. |



Copyright © 2024 Texas Instruments Incorporated

### 20.7.2.13 I2CPSC Register (Offset = Ch) [Reset = 0000h]

I2CPSC is shown in Figure 20-32 and described in Table 20-22.

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 20-32. 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 20-22. 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.<br>IPSC determines how much the CPU clock is divided to create the module clock of the I2C module:<br>module clock frequency = I2C input clock frequency/(IPSC + 1)<br>Note: IPSC must be initialized while the I2C module is in reset (IRS = 0 in I2CMDR).<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 20.7.2.14 I2CFFTX Register (Offset = 20h) [Reset = 0000h]

I2CFFTX is shown in Figure 20-33 and described in Table 20-23.

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 20-33. 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 20-23. I2CFFTX Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RESERVED | R | 0h | Reserved |
| 14 | I2CFFEN | R/W | 0h | I2C FIFO mode enable bit.<br>This bit must be enabled for either the transmit or the receive FIFO to operate correctly.<br>Reset type: SYSRSn<br>0h (R/W) = Disable the I2C FIFO mode.<br>1h (R/W) = Enable the I2C FIFO mode. |
| 13 | TXFFRST | R/W | 0h | Transmit FIFO Reset<br>Reset type: SYSRSn<br>0h (R/W) = Reset the transmit FIFO pointer to 0000 and hold the transmit FIFO in the reset state.<br>1h (R/W) = Enable the transmit FIFO operation. |
| 12-8 | TXFFST | R | 0h | Contains the status of the transmit FIFO:<br>xxxxx Transmit FIFO contains xxxxx bytes.<br>00000 Transmit FIFO is empty.<br>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.<br>Reset type: SYSRSn |
| 7 | TXFFINT | R | 0h | Transmit FIFO interrupt flag.<br>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.<br>Reset type: SYSRSn<br>0h (R/W) = Transmit FIFO interrupt condition has not occurred.<br>1h (R/W) = Transmit FIFO interrupt condition has occurred. |
| 6 | TXFFINTCLR | R-0/W1S | 0h | Transmit FIFO Interrupt Flag Clear<br>Reset type: SYSRSn<br>0h (R/W) = Writes of zeros have no effect. Reads return a 0.<br>1h (R/W) = Writing a 1 to this bit clears the TXFFINT flag. |
| 5 | TXFFIENA | R/W | 0h | Transmit FIFO Interrupt Enable<br>Reset type: SYSRSn<br>0h (R/W) = Disabled. TXFFINT flag does not generate an interrupt when set.<br>1h (R/W) = Enabled. TXFFINT flag does generate an interrupt when set. |



Copyright © 2024 Texas Instruments Incorporated

**Table 20-23. I2CFFTX Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 4-0 | TXFFIL | R/W | 0h | Transmit FIFO interrupt level.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 20.7.2.15 I2CFFRX Register (Offset = 21h) [Reset = 0000h]

I2CFFRX is shown in Figure 20-34 and described in Table 20-24.

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 20-34. 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 20-24. 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<br>Reset type: SYSRSn<br>0h (R/W) = Reset the receive FIFO pointer to 0000 and hold the receive FIFO in the reset state.<br>1h (R/W) = Enable the receive FIFO operation. |
| 12-8 | RXFFST | R | 0h | Contains the status of the receive FIFO:<br>xxxxx Receive FIFO contains xxxxx bytes<br>00000 Receive FIFO is empty.<br>Reset type: SYSRSn |
| 7 | RXFFINT | R | 0h | Receive FIFO interrupt flag.<br>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<br>Reset type: SYSRSn<br>0h (R/W) = Receive FIFO interrupt condition has not occurred.<br>1h (R/W) = Receive FIFO interrupt condition has occurred. |
| 6 | RXFFINTCLR | R-0/W1S | 0h | Receive FIFO interrupt flag clear bit.<br>Reset type: SYSRSn<br>0h (R/W) = Writes of zeros have no effect. Reads return a zero.<br>1h (R/W) = Writing a 1 to this bit clears the RXFFINT flag. |
| 5 | RXFFIENA | R/W | 0h | Receive FIFO interrupt enable bit.<br>Reset type: SYSRSn<br>0h (R/W) = Disabled. RXFFINT flag does not generate an interrupt when set.<br>1h (R/W) = Enabled. RXFFINT flag does generate an interrupt when set. |
| 4-0 | RXFFIL | R/W | 0h | Receive FIFO interrupt level.<br>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.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 20.7.3 I2C Registers to Driverlib Functions

**Table 20-25. I2C Registers to Driverlib Functions**

| File | Driverlib Function |
|---|---|
| **OAR** | |
| i2c.h | I2C_setOwnAddress |
| **IER** | |
| i2c.c | I2C_enableInterrupt |
| i2c.c | I2C_disableInterrupt |
| **STR** | |
| i2c.c | I2C_getInterruptStatus |
| i2c.c | I2C_clearInterruptStatus |
| i2c.h | I2C_isBusBusy |
| i2c.h | I2C_getStatus |
| i2c.h | I2C_clearStatus |
| **CLKL** | |
| i2c.c | I2C_initController |
| i2c.c | I2C_initControllerModuleFrequency |
| **CLKH** | |
| i2c.c | I2C_initController |
| i2c.c | I2C_initControllerModuleFrequency |
| **CNT** | |
| i2c.h | I2C_setDataCount |
| **DRR** | |
| i2c.h | I2C_getData |
| **TAR** | |
| i2c.h | I2C_setTargetAddress |
| **DXR** | |
| i2c.h | I2C_putData |
| **MDR** | |
| 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 |
| **ISRC** | |
| i2c.h | I2C_getInterruptSource |
| **EMDR** | |
| - | |
| **PSC** | |



Copyright © 2024 Texas Instruments Incorporated

**Table 20-25. 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 |
| **FFTX** | |
| 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 |
| **FFRX** | |
| 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 |



Copyright © 2024 Texas Instruments Incorporated

Chapter 21

# Multichannel Buffered Serial Port (McBSP)

This chapter describes the multichannel buffered serial port (McBSP).


**21.1 Introduction** ..... 2392
**21.2 Configuring Device Pins** ..... 2393
**21.3 McBSP Operation** ..... 2394
**21.4 McBSP Sample Rate Generator** ..... 2404
**21.5 McBSP Exception/Error Conditions** ..... 2411
**21.6 Multichannel Selection Modes** ..... 2420
**21.7 SPI Operation Using the Clock Stop Mode** ..... 2428
**21.8 Receiver Configuration** ..... 2435
**21.9 Transmitter Configuration** ..... 2457
**21.10 Emulation and Reset Considerations** ..... 2475
**21.11 Data Packing Examples** ..... 2478
**21.12 Interrupt Generation** ..... 2481
**21.13 McBSP Modes** ..... 2483
**21.14 Special Case: External Device is the Transmit Frame Master** ..... 2483
**21.15 Software** ..... 2485
**21.16 McBSP Registers** ..... 2485





Copyright © 2024 Texas Instruments Incorporated

## 21.1 Introduction

This device provides up to two high-speed multichannel buffered serial ports (McBSPs) that allow direct interface to codecs and other devices in a system. The McBSP consists of a data-flow path and a control path connected to external devices by six pins as shown in Figure 21-1.

Data is communicated to devices interfaced with the McBSP via the data transmit (DX) pin for transmission and via the data receive (DR) pin for reception. Control information in the form of clocking and frame synchronization is communicated via the following pins: CLKX (transmit clock), CLKR (receive clock), FSX (transmit frame synchronization), and FSR (receive frame synchronization).

The CPU and the DMA controller communicate with the McBSP through 16-bit wide registers accessible via the internal peripheral bus. The CPU or the DMA controller writes the data to be transmitted to the data transmit registers (DXR1, DXR2). Data written to the DXRs is shifted out to the DX pin via the transmit shift registers (XSR1, XSR2). Similarly, receive data on the DR pin is shifted into the receive shift registers (RSR1, RSR2) and copied into the receive buffer registers (RBR1, RBR2). The contents of the receive buffer registers (RBRs) is then copied to the data receive registers (DRRs), which can be read by the CPU or the DMA controller. This allows simultaneous movement of internal and external data communications.

If the serial word length is 8 bits, 12 bits, or 16 bits, the DRR2, RBR2, RSR2, DXR2, and XSR2 registers are not used (written, read, or shifted) For larger word lengths, these registers are needed to hold the most significant bits.

The frame and clock loop-back is implemented at chip level to enable CLKX and FSX to drive CLKR and FSR. If the loop-back is enabled, the CLKR and FSR get their signals from the CLKX and FSX pads; instead of the CLKR and FSR pins.

### *21.1.1 MCBSP Related Collateral*

**Foundational Materials**

- C2000 Academy - McBSP
- *KeyStone Architecture Multichannel Buffered Serial Port (McBSP)*

### *21.1.2 Features of the McBSPs*

The McBSPs feature:

- Full-duplex communication
- Double-buffered transmission and triple-buffered reception, allowing a continuous data stream
- Independent clocking and framing for reception and transmission
- The capability to send interrupts to the CPU and to send DMA events to the DMA controller
- 128 channels for transmission and reception
- Multichannel selection modes that enable or disable block transfers in each of the channels
- Direct interface to industry-standard codecs, analog interface chips (AICs), and other serially connected A/D and D/A devices
- Support for external generation of clock signals and frame-synchronization signals
- A programmable sample rate generator for internal generation and control of clock signals and frame-synchronization signals
- Programmable polarity for frame-synchronization pulses and clock signals
- Direct interface to:
  - T1/E1 framers
  - IOM-2 compliant devices
  - AC97-compliant devices (the necessary multiphase frame capability is provided)
  - I2S compliant devices
  - SPI devices



Copyright © 2024 Texas Instruments Incorporated

- A wide selection of data sizes: 8, 12, 16, 20, 24, and 32 bits

**Note**

A value of the chosen data size is referred to as a *serial word* or *word* throughout the McBSP chapter. Elsewhere, *word* is used to describe a 16-bit value.

- µ-law and A-law companding
- The option of transmitting/receiving 8-bit data with the LSB first
- Status bits for flagging exception/error conditions
- ABIS mode is not supported

### *21.1.3 McBSP Pins/Signals*

Table 21-1 describes the McBSP interface pins and some internal signals.

**Table 21-1. McBSP Interface Pins/Signals**

| McBSP-A Pin | McBSP-B Pin | Type | Description |
|---|---|---|---|
| MCLKRA | MCLKRB | I/O | Supplies or reflects the receive clock; supplies the input clock of the sample rate generator |
| MCLKXA | MCLKXB | I/O | Supplies or reflects the transmit clock; supplies the input clock of the sample rate generator |
| MDRA | MDRB | I | Serial data receive pin |
| MDXA | MDXB | O | Serial data transmit pin |
| MFSRA | MFSRB | I/O | Supplies or reflects the receive frame-sync signal; controls sample rate generator synchronization when GSYNC = 1 (see Section 21.4.3) |
| MFSXA | MFSXB | I/O | Supplies or reflects the transmit frame-sync signal |
| **CPU Interrupt Signals** | | | |
| MRINT | | | Receive interrupt to CPU |
| MXINT | | | Transmit interrupt to CPU |
| **DMA Events** | | | |
| REVT | | | Receive synchronization event to DMA |
| XEVT | | | Transmit synchronization event to DMA |

#### 21.1.3.1 McBSP Generic Block Diagram

The McBSP consists of a data-flow path and a control path connected to external devices by six pins as shown in Figure 21-1. The figure and the text in this section use generic pin names.

## 21.2 Configuring Device Pins

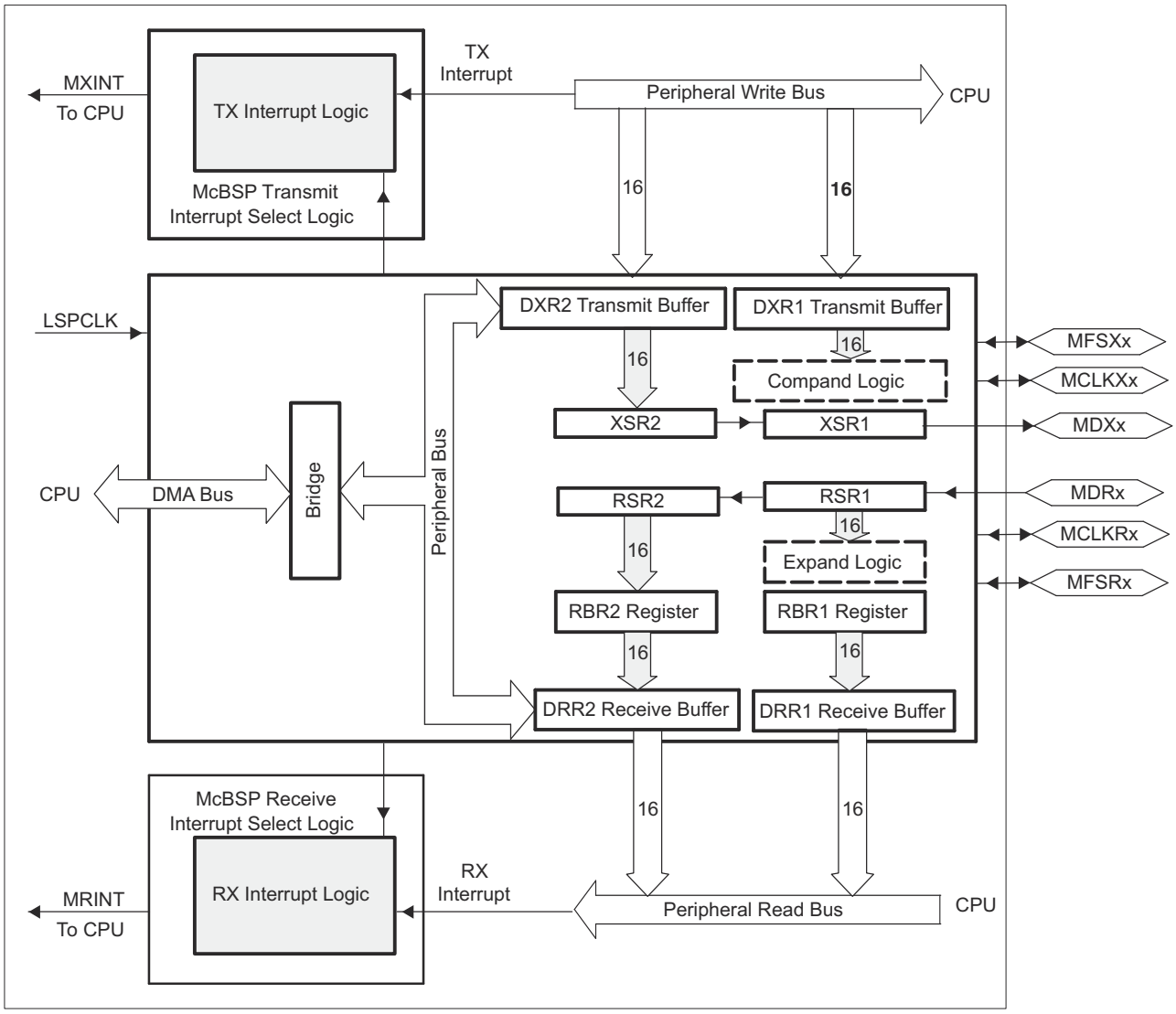
The GPIO mux registers must be configured to connect this peripheral to the device pins. To avoid glitches on the pins, the GPyGMUX bits must be configured first (while keeping the corresponding GPyMUX bits at the default of zero), followed by writing the GPyMUX register to the desired value.

Some IO functionality is defined by GPIO register settings independent of this peripheral. For input signals, the GPIO input qualification 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.



Copyright © 2024 Texas Instruments Incorporated




A. Not available in all devices. See the device-specific data sheet.

**Figure 21-1. Conceptual Block Diagram of the McBSP**

## 21.3 McBSP Operation

This section addresses the following topics:

- Data transfer process
- Companding (compressing and expanding) data
- Clocking and framing data
- Frame phases
- McBSP reception
- McBSP transmission
- Interrupts and DMA events generated by McBSPs



Copyright © 2024 Texas Instruments Incorporated

### 21.3.1 Data Transfer Process of McBSPs

Figure 21-2 shows a diagram of the McBSP data transfer paths. The McBSP receive operation is triple-buffered, and transmit operation is double-buffered. The use of registers varies, depending on whether the defined length of each serial word is 16 bits.

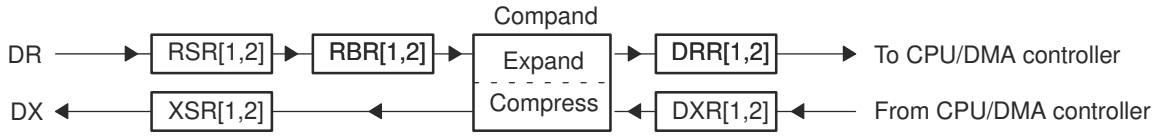



**Figure 21-2. McBSP Data Transfer Paths**

#### 21.3.1.1 Data Transfer Process for Word Length of 8, 12, or 16 Bits

If the word length is 16 bits or smaller, only one 16-bit register is needed at each stage of the data transfer paths. The registers DRR2, RBR2, RSR2, DXR2, and XSR2 are not used (written, read, or shifted).

Receive data arrives on the DR pin and is shifted into receive shift register 1 (RSR1). Once a full word is received, the content of RSR1 is copied to the receive buffer register 1 (RBR1), that is, if RBR1 is not full with previous data. RBR1 is then copied to the data receive register 1 (DRR1), unless the previous content of DRR1 has not been read by the CPU or the DMA controller. If the companding feature of the McBSP is implemented, the required word length is 8 bits and receive data is expanded into the appropriate format before being passed from RBR1 to DRR1. For more details about reception, see Section 21.3.5.

Transmit data is written by the CPU or the DMA controller to the data transmit register 1 (DXR1). If there is no previous data in the transmit shift register (XSR1), the value in DXR1 is copied to XSR1; otherwise, DXR1 is copied to XSR1 when the last bit of the previous data is shifted out on the DX pin. If selected, the companding module compresses 16-bit data into the appropriate 8-bit format before passing it to XSR1. After transmit frame synchronization, the transmitter begins shifting bits from XSR1 to the DX pin. For more details about transmission, see Section 21.3.6.

#### 21.3.1.2 Data Transfer Process for Word Length of 20, 24, or 32 Bits

If the word length is larger than 16 bits, two 16-bit registers are needed at each stage of the data transfer paths. The registers DRR2, RBR2, RSR2, DXR2, and XSR2 are needed to hold the most significant bits.

Receive data arrives on the DR pin and is shifted first into RSR2 and then into RSR1. Once the full word is received, the contents of RSR2 and RSR1 are copied to RBR2 and RBR1, respectively, if RBR1 is not full. Then the contents of RBR2 and RBR1 are copied to DRR2 and DRR1, respectively, unless the previous content of DRR1 has not been read by the CPU or the DMA controller. The CPU or the DMA controller must read data from DRR2 first and then from DRR1. When DRR1 is read, the next RBR-to-DRR copy occurs. For more details about reception, see Section 21.3.5.

For transmission, the CPU or the DMA controller must write data to DXR2 first and then to DXR1. When new data arrives in DXR1, if there is no previous data in XSR1, the contents of DXR2 and DXR1 are copied to XSR2 and XSR1, respectively; otherwise, the contents of the DXRs are copied to the XSRs when the last bit of the previous data is shifted out on the DX pin. After transmit frame synchronization, the transmitter begins shifting bits from the XSRs to the DX pin. For more details about transmission, see Section 21.3.6.

### 21.3.2 Companding (Compressing and Expanding) Data

Companding (COMpressing and exPANDing) hardware allows compression and expansion of data in either μ-law or A-law format. The companding standard employed in the United States and Japan is μ-law. The European companding standard is referred to as A-law. The specifications for μ-law and A-law log PCM are part of the CCITT G.711 recommendation.

A-law and μ-law allow 13 bits and 14 bits of dynamic range, respectively. Any values outside this range are set to the most positive or most negative value. Thus, for companding to work best, the data transferred to and from the McBSP via the CPU or DMA controller must be at least 16 bits wide.



Copyright © 2024 Texas Instruments Incorporated

The μ-law and A-law formats both encode data into 8-bit code words. Companded data is always 8 bits wide; the appropriate word length bits (RWDLEN1, RWDLEN2, XWDLEN1, XWDLEN2) must therefore be set to 0, indicating an 8-bit wide serial data stream. If companding is enabled and either of the frame phases does not have an 8-bit word length, companding continues as if the word length is 8 bits.

Figure 21-3 illustrates the companding processes. When companding is chosen for the transmitter, compression occurs during the process of copying data from DXR1 to XSR1. The transmit data is encoded according to the specified companding law (A-law or μ-law). When companding is chosen for the receiver, expansion occurs during the process of copying data from RBR1 to DRR1. The receive data is decoded to twos complement format.

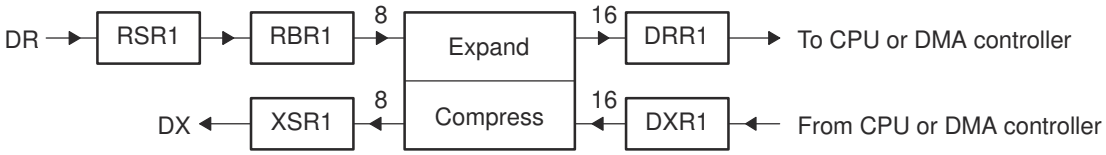



**Figure 21-3. Companding Processes**

#### 21.3.2.1 Companding Formats

For reception, the 8-bit compressed data in RBR1 is expanded to left-justified 16-bit data in DRR1. The receive sign-extension and justification mode specified in RJUST is ignored when companding is used.

For transmission using μ-law compression, the 14 data bits must be left-justified in DXR1 with the remaining two low-order bits filled with 0s as shown in Figure 21-4.

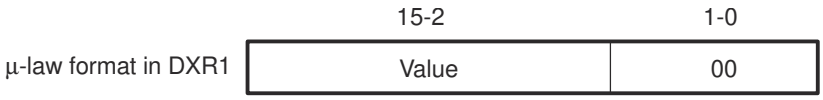



**Figure 21-4. μ-Law Transmit Data Companding Format**

For transmission using A-law compression, the 13 data bits must be left-justified in DXR1, with the remaining three low-order bits filled with 0s as shown in Figure 21-5.

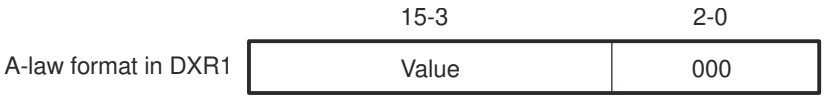



**Figure 21-5. A-Law Transmit Data Companding Format**

#### 21.3.2.2 Capability to Compand Internal Data

If the McBSP is unused (the serial port transmit and receive sections are reset), the companding hardware can compand internal data. This can be used to:

- Convert linear to the appropriate μ-law or A-law format
- Convert μ-law or A-law to the linear format
- Observe the quantization effects in companding by transmitting linear data and compressing and re-expanding this data. This is useful only if both XCOMPAND and RCOMPAND enable the same companding format.

Figure 21-6 shows two methods by which the McBSP can compand internal data. Data paths for these two methods are used to indicate:

- When both the transmit and receive sections of the serial port are reset, DRR1 and DXR1 are connected internally through the companding logic. Values from DXR1 are compressed, as selected by XCOMPAND, and then expanded, as selected by RCOMPAND. RRDY and XRDY bits are not set. However, data is available in DRR1 within four CPU clocks after being written to DXR1.



Copyright © 2024 Texas Instruments Incorporated

The advantage of this method is the speed. The disadvantage is that there is no synchronization available to the CPU and DMA to control the flow. DRR1 and DXR1 are internally connected if the (X/R)COMPAND bits are set to 10b or 11b (compand using μ-law or A-law).

- The McBSP is enabled in digital loopback mode with companding appropriately enabled by RCOMPAND and XCOMPAND. Receive and transmit interrupts (RINT when RINTM = 0 and XINT when XINTM = 0) or synchronization events (REVT and XEVT) allow synchronization of the CPU or DMA to these conversions, respectively. Here, the time for this companding depends on the serial bit rate selected.

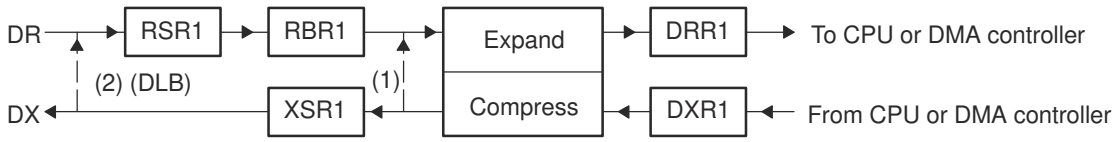



**Figure 21-6. Two Methods by Which the McBSP Can Compand Internal Data**

#### 21.3.2.3 Reversing Bit Order: Option to Transfer LSB First

Generally, the McBSP transmits or receives all data with the most significant bit (MSB) first. However, certain 8-bit data protocols (that do not use companded data) require the least significant bit (LSB) to be transferred first. If you set XCOMPAND = 01b in XCR2, the bit ordering of 8-bit words is reversed (LSB first) before being sent from the serial port. If you set RCOMPAND = 01b in RCR2, the bit ordering of 8-bit words is reversed during reception. Similar to companding, this feature is enabled only if the appropriate word length bits are set to 0, indicating that 8-bit words are to be transferred serially. If either phase of the frame does not have an 8-bit word length, the McBSP assumes the word length is eight bits, and LSB-first ordering is done.

### *21.3.3 Clocking and Framing Data*

This section explains basic concepts and terminology important for understanding how McBSP data transfers are timed and delimited.

#### 21.3.3.1 Clocking

**Note**

The McBSP cannot operate at a frequency faster than ½ the LSPCLK frequency. When driving CLKX or CLKR at the pin, choose an appropriate input clock frequency. When using the internal sample rate generator for CLKX and/or CLKR, choose an appropriate input clock frequency and divide down value (CLKDV) (that is, be certain that CLKX or CLKR ≤ LSPCLK/2).

Data is shifted one bit at a time from the DR pin to the RSRs or from the XSRs to the DX pin. The time for each bit transfer is controlled by the rising or falling edge of a clock signal.

The receive clock signal (CLKR) controls bit transfers from the DR pin to the RSRs. The transmit clock signal (CLKX) controls bit transfers from the XSRs to the DX pin. CLKR or CLKX can be derived from a pin at the boundary of the McBSP or derived from inside the McBSP. The polarities of CLKR and CLKX are programmable.

Figure 21-7 shows how the clock signal controls the timing of each bit transfer on the pin.

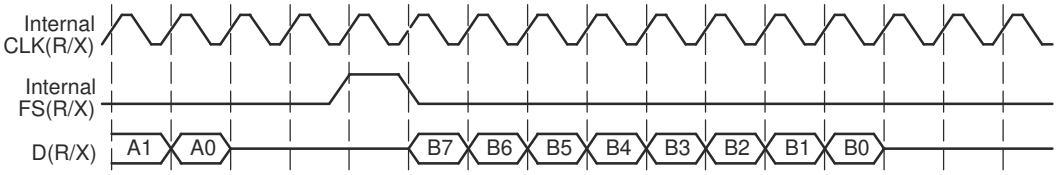



**Figure 21-7. Example - Clock Signal Control of Bit Transfer Timing**



Copyright © 2024 Texas Instruments Incorporated

#### 21.3.3.2 Serial Words

Bits traveling between a shift register (RSR or XSR) and a data pin (DR or DX) are transferred in a group called a serial word. You can define how many bits are in a word.

Bits coming in on the DR pin are held in RSR until RSR holds a full serial word. Only then is the word passed to RBR (and ultimately to the DRR).

During transmission, XSR does not accept new data from DXR until a full serial word has been passed from XSR to the DX pin.

In the exmaple in Figure 21-7, an 8-bit word size was defined (see bits 7 through 0 of word B being transferred).

#### 21.3.3.3 Frames and Frame Synchronization

One or more words are transferred in a group called a frame. You can define how many words are in a frame.

All of the words in a frame are sent in a continuous stream. However, there can be pauses between frame transfers. The McBSP uses frame-synchronization signals to determine when each frame is received/transmitted. When a pulse occurs on a frame-synchronization signal, the McBSP begins receiving/transmitting a frame of data. When the next pulse occurs, the McBSP receives/transmits the next frame, and so on.

Pulses on the receive frame-synchronization (FSR) signal initiate frame transfers on the DR data pin. Pulses on the transmit frame-sync (FSX) signal initiate frame transfers on DX. FSR or FSX can be derived from a pin at the boundary of the McBSP or derived from inside the McBSP.

In Figure 21-7, a one-word frame is transferred when a frame-synchronization pulse occurs.

In McBSP operation, the inactive-to-active transition of the frame-synchronization signal indicates the start of the next frame. For this reason, the frame-synchronization signal may be high for an arbitrary number of clock cycles. Only after the signal is recognized to have gone inactive, and then active again, does the next frame synchronization occur.

#### 21.3.3.4 Generating Transmit and Receive Interrupts

The McBSP can send receive and transmit interrupts to the CPU to indicate specific events in the McBSP. To facilitate detection of frame synchronization, these interrupts can be sent in response to frame-synchronization pulses. Set the appropriate interrupt mode bits to 10b (for reception, RINTM = 10b; for transmission, XINTM = 10b).

##### *21.3.3.4.1 Detecting Frame-Synchronization Pulses, Even in Reset State*

Unlike other serial port interrupt modes, this mode can operate while the associated portion of the serial port is in reset (such as activating RINT when the receiver is in reset). In this case, FSRM/FSXM and FSRP/FSXP still select the appropriate source and polarity of frame synchronization. Thus, even when the serial port is in the reset state, these signals are synchronized to the CPU clock and then sent to the CPU in the form of RINT and XINT at the point at which they feed the receiver and transmitter of the serial port. Consequently, a new frame-synchronization pulse can be detected, and after this occurs the CPU can take the serial port out of reset safely.

#### 21.3.3.5 Ignoring Frame-Synchronization Pulses

The McBSP can be configured to ignore transmit and/or receive frame-synchronization pulses. To have the receiver or transmitter recognize frame-synchronization pulses, clear the appropriate frame-synchronization ignore bit (RFIG = 0 for the receiver, XFIG = 0 for the transmitter). To have the receiver or transmitter ignore frame-synchronization pulses until the desired frame length or number of words is reached, set the appropriate frame-synchronization ignore bit (RFIG = 1 for the receiver, XFIG = 1 for the transmitter). For more details on unexpected frame-synchronization pulses, see one of the following topics:

- *Unexpected Receive Frame-Synchronization Pulse* (see Section 21.5.3)
- *Unexpected Transmit Frame-Synchronization Pulse* (see Section 21.5.6)



Copyright © 2024 Texas Instruments Incorporated

You can also use the frame-synchronization ignore function for data packing (for more details, see Section 21.11.2).

#### 21.3.3.6 Frame Frequency

The frame frequency is determined by the period between frame-synchronization pulses and is defined by the equation:

$$\text{Frame Frequency} = \frac{\text{Clock Frequency}}{\text{Number of Clock Cycles Between Frame- Sync Pulses}}$$

The frame frequency can be increased by decreasing the time between frame-synchronization pulses (limited only by the number of bits per frame). As the frame transmit frequency increases, the inactivity period between the data packets for adjacent transfers decreases to zero.

#### 21.3.3.7 Maximum Frame Frequency

The minimum number of clock cycles between frame synchronization pulses is equal to the number of bits transferred per frame. The maximum frame frequency is defined by the equation:

$$\text{Maximum Frame Frequency} = \frac{\text{Clock Frequency}}{\text{Number of Bits Per Frame}}$$

Figure 21-8 shows the McBSP operating at maximum packet frequency. At maximum packet frequency, the data bits in consecutive packets are transmitted contiguously with no inactivity between bits.

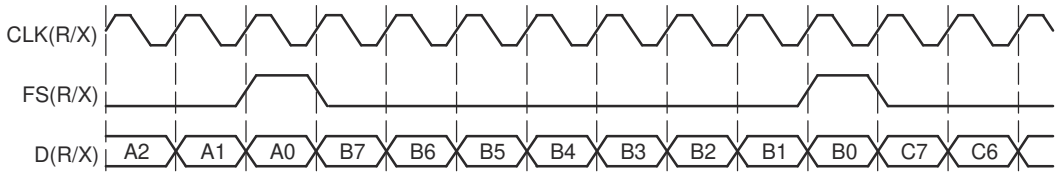



**Figure 21-8. McBSP Operating at Maximum Packet Frequency**

If there is a 1-bit data delay as shown in Figure 21-8, the frame-synchronization pulse overlaps the last bit transmitted in the previous frame. Effectively, this permits a continuous stream of data, making frame-synchronization pulses redundant. Theoretically, only an initial frame-synchronization pulse is required to initiate a multipacket transfer.

The McBSP supports operation of the serial port in this fashion by ignoring the successive frame-synchronization pulses. Data is clocked into the receiver or clocked out of the transmitter during every clock cycle.

**Note**

For XDATDLY = 0 (0-bit data delay), the first bit of data is transmitted asynchronously to the internal transmit clock signal (CLKX). For more details, see Section 21.9.13.

### *21.3.4 Frame Phases*

The McBSP allows you to configure each frame to contain one or two phases. The number of words and the number of bits per word can be specified differently for each of the two phases of a frame, allowing greater flexibility in structuring data transfers. For example, you might define a frame as consisting of one phase containing two words of 16 bits each, followed by a second phase consisting of 10 words of 8 bits each. This configuration permits you to compose frames for custom applications or, in general, to maximize the efficiency of data transfers.



Copyright © 2024 Texas Instruments Incorporated

#### 21.3.4.1 Number of Phases, Words, and Bits Per Frame

Table 21-2 shows which bit-fields in the receive control registers (RCR1 and RCR2) and in the transmit control registers (XCR1 and XCR2) determine the number of phases per frame, the number of words per frame, and number of bits per word for each phase, for the receiver and transmitter. The maximum number of words per frame is 128 for a single-phase frame and 256 for a dual-phase frame. The number of bits per word can be 8, 12, 16, 20, 24, or 32 bits.

**Table 21-2. Register Bits That Determine the Number of Phases, Words, and Bits**

| Operation | Number of Phases | Words per Frame Set With | Bits per Word Set With |
|---|---|---|---|
| Reception | 1 (RPHASE = 0) | RFRLEN1 | RWDLEN1 |
| Reception | 2 (RPHASE = 1) | RFRLEN1 and RFRLEN2 | RWDLEN1 for phase 1 |
| | | | RWDLEN2 for phase 2 |
| Transmission | 1 (XPHASE = 0) | XFRLEN1 | XWDLEN1 |
| Transmission | 2 (XPHASE = 1) | XFRLEN1 and XFRLEN2 | XWDLEN1 for phase 1 |
| | | | XWDLEN2 for phase 2 |

#### 21.3.4.2 Single-Phase Frame Example

Figure 21-9 shows an example of a single-phase data frame containing one 8-bit word. Because the transfer is configured for one data bit delay, the data on the DX and DR pins are available one clock cycle after FS(R/X) goes active. The figure makes the following assumptions:

- (R/X)PHASE = 0: Single-phase frame
- (R/X)FRLEN1 = 0b: 1 word per frame
- (R/X)WDLEN1 = 000b: 8-bit word length
- (R/X)FRLEN2 and (R/X)WDLEN2 are ignored
- CLK(X/R)P = 0: Receive data clocked on falling edge; transmit data clocked on rising edge
- FS(R/X)P = 0: Active-high frame-synchronization signals
- (R/X)DATDLY = 01b: 1-bit data delay

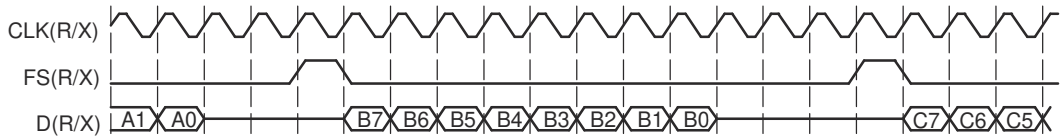
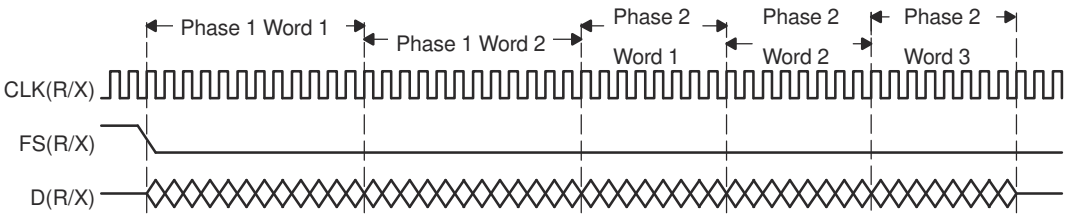



**Figure 21-9. Single-Phase Frame for a McBSP Data Transfer**

#### 21.3.4.3 Dual-Phase Frame Example

Figure 21-10 shows an example of a frame where the first phase consists of two words of 12 bits each, followed by a second phase of three words of 8 bits each. The entire bit stream in the frame is contiguous. There are no gaps either between words or between phases.




A. XRDY gets asserted once per phase. So, if there are 2 phases, XRDY gets asserted twice (once per phase).

**Figure 21-10. Dual-Phase Frame for a McBSP Data Transfer**



Copyright © 2024 Texas Instruments Incorporated

#### 21.3.4.4 Implementing the AC97 Standard With a Dual-Phase Frame

Figure 21-11 shows an example of the Audio Codec '97 (AC97) standard, which uses the dual-phase frame feature. Notice that words, not individual bits, are shown on the D(R/X) signal. The first phase (P1) consists of a single 16-bit word. The second phase (P2) consists of twelve 20-bit words. The phase configurations are listed after the figure.

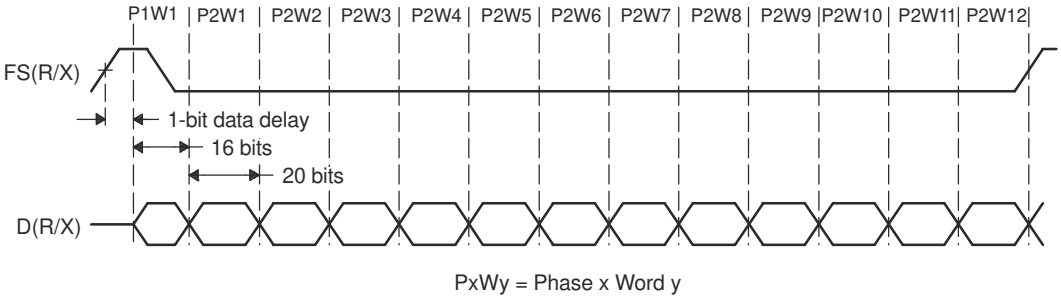



**Figure 21-11. Implementing the AC97 Standard With a Dual-Phase Frame**

- (R/X)PHASE = 1: Dual-phase frame
- (R/X)FRLEN1 = 0000000b: 1 word in phase 1
- (R/X)WDLEN1 = 010b: 16 bits per word in phase 1
- (R/X)FRLEN2 = 0001011b: 12 words in phase 2
- (R/X)WDLEN2 = 011b: 20 bits per word in phase 2
- CLKRP/CLKXP= 0: Receive data sampled on falling edge of internal CLKR / transmit data clocked on rising edge of internal CLKX
- FSRP/FSXP = 0: Active-high frame-sync signal
- (R/X)DATDLY = 01b: Data delay of 1 clock cycle (1-bit data delay)

Figure 21-12 shows the timing of an AC97-standard data transfer near frame synchronization. In this figure, individual bits are shown on D(R/X). Specifically, it shows the last two bits of phase 2 of one frame and the first four bits of phase 1 of the next frame. Regardless of the data delay, data transfers can occur without gaps. The first bit of the second frame (P1W1B15) immediately follows the last bit of the first frame (P2W12B0). Because a 1-bit data delay has been chosen, the transition on the frame-sync signal can occur when P2W12B0 is transferred.

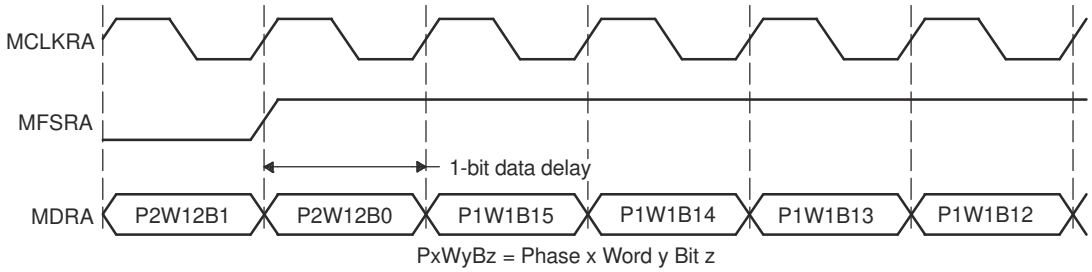



**Figure 21-12. Timing of an AC97-Standard Data Transfer Near Frame Synchronization**

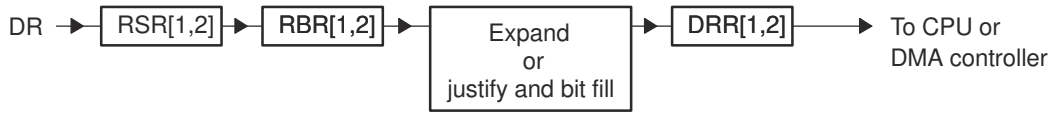


Copyright © 2024 Texas Instruments Incorporated

### 21.3.5 McBSP Reception

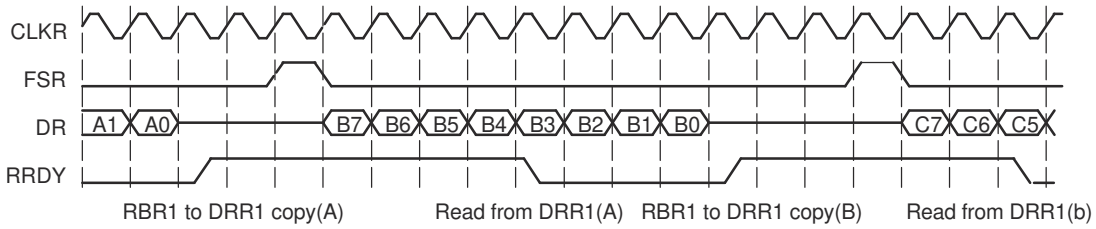
This section explains the fundamental process of reception in the McBSP. For details about how to program the McBSP receiver, see Section 21.8.

Figure 21-13 and Figure 21-14 show how reception occurs in the McBSP. Figure 21-13 shows the physical path for the data. Figure 21-14 is a timing diagram showing signal activity for one possible reception scenario. A description of the process follows the figures.




A. RSR[1,2]: Receive shift registers 1 and 2
B. RBR[1,2]: Receive buffer registers 1 and 2
C. DRR[1,2]: Data receive registers 1 and 2

**Figure 21-13. McBSP Reception Physical Data Path**




A. CLKR: Internal receive clock
B. FSR: Internal receive frame-synchronization signal
C. DR: Data on DR pin
D. RRDY: Status of receiver ready bit (high is 1)

**Figure 21-14. McBSP Reception Signal Activity**

The following process describes how data travels from the DR pin to the CPU or to the DMA controller:

1. The McBSP waits for a receive frame-synchronization pulse on internal FSR.
2. When the pulse arrives, the McBSP inserts the appropriate data delay that is selected with the RDATDLY bits of RCR2.
   In the preceding timing diagram, a 1-bit data delay is selected.
3. The McBSP accepts data bits on the DR pin and shifts them into the receive shift registers.
   If the word length is 16 bits or smaller, only RSR1 is used. If the word length is larger than 16 bits, RSR2 and RSR1 are used and RSR2 contains the most-significant bits. For details on choosing a word length, see Section 21.8.8.
4. When a full word is received, the McBSP copies the contents of the receive shift registers to the receive buffer registers, provided that RBR1 is not full with previous data.
   If the word length is 16 bits or smaller, only RBR1 is used. If the word length is larger than 16 bits, RBR2 and RBR1 are used and RBR2 contains the most-significant bits.
5. The McBSP copies the contents of the receive buffer registers into the data receive registers, provided that DRR1 is not full with previous data. When DRR1 receives new data, the receiver ready bit (RRDY) is set in SPCR1. This indicates that received data is ready to be read by the CPU or the DMA controller.
   If the word length is 16 bits or smaller, only DRR1 is used. If the word length is larger than 16 bits, DRR2 and DRR1 are used and DRR2 contains the most-significant bits.
   If companding is used during the copy (RCOMPAND = 10b or 11b in RCR2), the 8-bit compressed data in RBR1 is expanded to a left-justified 16-bit value in DRR1. If companding is disabled, the data copied from RBR[1,2] to DRR[1,2] is justified and bit filled according to the RJUST bits.



Copyright © 2024 Texas Instruments Incorporated

6. The CPU or the DMA controller reads the data from the data receive registers. When DRR1 is read, RRDY is cleared and the next RBR-to-DRR copy is initiated.

**Note**

If both DRRs are required (word length larger than 16 bits), the CPU or the DMA controller must read from DRR2 first and then from DRR1. As soon as DRR1 is read, the next RBR-to-DRR copy is initiated. If DRR2 is not read first, the data in DRR2 is lost.

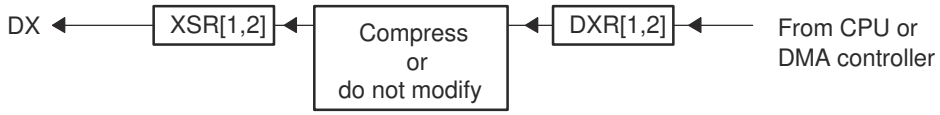
When activity is not properly timed, errors can occur. See the following topics for more details:

- Section 21.5.2
- Section 21.5.3

### *21.3.6 McBSP Transmission*

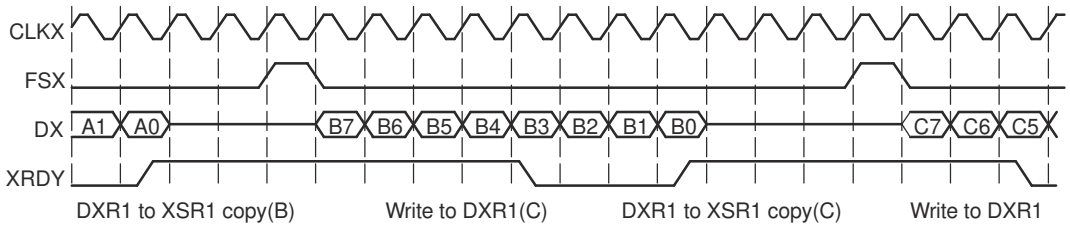
This section explains the fundamental process of transmission in the McBSP. For details about how to program the McBSP transmitter, see Section 21.9.

Figure 21-15 and Figure 21-16 show how transmission occurs in the McBSP. Figure 21-15 shows the physical path for the data. Figure 21-16 is a timing diagram showing signal activity for one possible transmission scenario. A description of the process follows the figures.




A. XSR[1,2]: Transmit shift registers 1 and 2
B. DXR[1,2]: Data transmit registers 1 and 2

**Figure 21-15. McBSP Transmission Physical Data Path**




A. CLKX: Internal transmit clock
B. FSX: Internal transmit frame-synchronization signal
C. DX: Data on DX pin
D. XRDY: Status of transmitter ready bit (high is 1)

**Figure 21-16. McBSP Transmission Signal Activity**

1. The CPU or the DMA controller writes data to the data transmit registers. When DXR1 is loaded, the transmitter ready bit (XRDY) is cleared in SPCR2 to indicate that the transmitter is not ready for new data. If the word length is 16 bits or smaller, only DXR1 is used. If the word length is larger than 16 bits, DXR2 and DXR1 are used and DXR2 contains the most-significant bits. For details on choosing a word length, see Section 21.9.9.

**Note**

If both DXRs are needed (word length larger than 16 bits), the CPU or the DMA controller must load DXR2 first and then load DXR1. As soon as DXR1 is loaded, the contents of both DXRs are copied to the transmit shift registers (XSRs), as described in the next step. If DXR2 is not loaded first, the previous content of DXR2 is passed to the XSR2.



Copyright © 2024 Texas Instruments Incorporated

2. When new data arrives in DXR1, the McBSP copies the content of the data transmit registers to the transmit shift registers. In addition, the transmit ready bit (XRDY) is set. This indicates that the transmitter is ready to accept new data from the CPU or the DMA controller.
   If the word length is 16 bits or smaller, only XSR1 is used. If the word length is larger than 16 bits, XSR2 and XSR1 are used and XSR2 contains the most-significant bits.
   If companding is used during the transfer (XCOMPAND = 10b or 11b in XCR2), the McBSP compresses the 16-bit data in DXR1 to 8-bit data in the μ-law or A-law format in XSR1. If companding is disabled, the McBSP passes data from the DXRs to the XSRs without modification.
3. The McBSP waits for a transmit frame-synchronization pulse on internal FSX.
4. When the pulse arrives, the McBSP inserts the appropriate data delay that is selected with the XDATDLY bits of XCR2.
   In the preceding timing diagram (Figure 21-16), a 1-bit data delay is selected.
5. The McBSP shifts data bits from the transmit shift registers to the DX pin.

When activity is not properly timed, errors can occur. See the following topics for more details:

- Section 21.5.4
- Section 21.5.5
- Section 21.5.6

### *21.3.7 Interrupts and DMA Events Generated by a McBSP*

The McBSP sends notification of important events to the CPU and DMA via the internal signals shown in Table 21-3.

**Table 21-3. Interrupts and DMA Events Generated by a McBSP**

| Internal Signal | Description |
|---|---|
| RINT | Receive interrupt<br>The McBSP sends a receive interrupt request to the CPU based upon a selected condition in the receiver of the McBSP (a condition selected by the RINTM bits of SPCR1). |
| XINT | Transmit interrupt<br>The McBSP sends a transmit interrupt request to the CPU based upon a selected condition in the transmitter of the McBSP (a condition selected by the XINTM bits of SPCR2). |
| REVT | Receive synchronization event<br>An REVT signal is sent to the DMA when data has been received in the data receive registers (DRRs). |
| XEVT | Transmit synchronization event<br>An XEVT signal is sent to the DMA when the data transmit registers (DXRs) are ready to accept the next serial word for transmission. |

## 21.4 McBSP Sample Rate Generator

Each McBSP contains a sample rate generator (SRG) that can be programmed to generate an internal data clock (CLKG) and an internal frame-synchronization signal (FSG). CLKG can be used for bit shifting on the data receive (DR) pin and/or the data transmit (DX) pin. FSG can be used to initiate frame transfers on DR and/or DX. Figure 21-17 is a conceptual block diagram of the sample rate generator.



Copyright © 2024 Texas Instruments Incorporated

### 21.4.1 Block Diagram

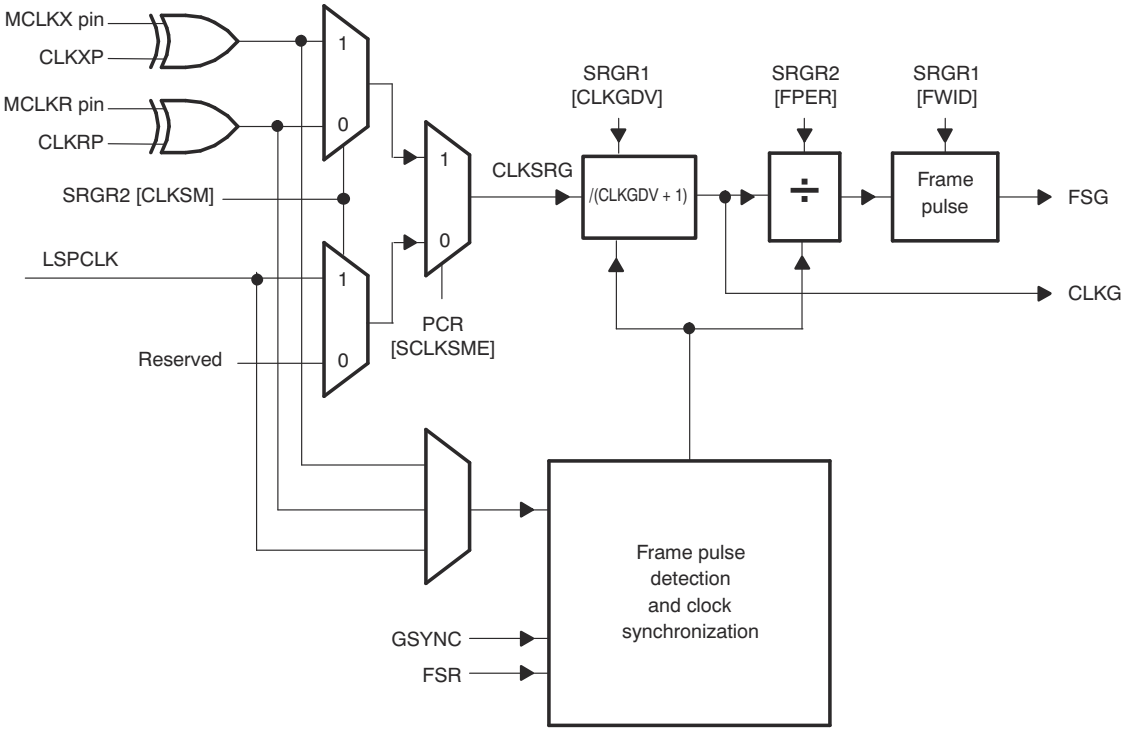



**Figure 21-17. Conceptual Block Diagram of the Sample Rate Generator**

The source clock for the sample rate generator (labeled CLKSRG in the diagram) can be supplied by the LSPCLK, or by an external pin (MCLKX or MCLKR ). The source is selected with the SCLKME bit of PCR and the CLKSM bit of SRGR2. If a pin is used, the polarity of the incoming signal can be inverted with the appropriate polarity bit (CLKXP of PCR or CLKRP of PCR).

The sample rate generator has a three-stage clock divider that gives CLKG and FSG programmability. The three stages provide:

- Clock divide-down. The source clock is divided according to the CLKGDV bits of SRGR1 to produce CLKG.
- Frame period divide-down. CLKG is divided according to the FPER bits of SRGR2 to control the period from the start of a frame-pulse to the start of the next pulse.
- Frame-synchronization pulse-width countdown. CLKG cycles are counted according to the FWID bits of SRGR1 to control the width of each frame-synchronization pulse.

**Note**

The McBSP cannot operate at a frequency faster than ½ the source clock frequency. You must choose an input clock frequency and a CLKGDV value such that CLKG is less than or equal to ½ the source clock frequency.

In addition to the three-stage clock divider, the sample rate generator has a frame-synchronization pulse detection and clock synchronization module that allows synchronization of the clock divide down with an incoming frame-synchronization pulse on the FSR pin. This feature is enabled or disabled with the GSYNC bit of SRGR2.

For details on getting the sample rate generator ready for operation, see Section 21.4.4.



Copyright © 2024 Texas Instruments Incorporated

#### 21.4.1.1 Clock Generation in the Sample Rate Generator

The sample rate generator can produce a clock signal (CLKG) for use by the receiver, the transmitter, or both. Use of the sample rate generator to drive clocking is controlled by the clock mode bits (CLKRM and CLKXM) in the pin control register (PCR). When a clock mode bit is set to 1 (CLKRM = 1 for reception, CLKXM = 1 for transmission), the corresponding data clock (CLKR for reception, CLKX for transmission) is driven by the internal sample rate generator output clock (CLKG).

The effects of CLKRM = 1 and CLKXM = 1 on the McBSP are partially affected by the use of the digital loopback mode and the clock stop (SPI) mode, respectively, as described in Table 21-4. The digital loopback mode (described in Section 21.8.4) is selected with the DLB bit of SPCR1. The clock stop mode (described in Section 21.7.2) is selected with the CLKSTP bits of SPCR1.

When using the sample rate generator as a clock source, make sure the sample rate generator is enabled (GRST = 1).

**Table 21-4. Effects of DLB and CLKSTP on Clock Modes**

| Mode Bit Settings | | Effect |
|---|---|---|
| CLKRM = 1 | DLB = 0<br>(Digital loopback mode disabled) | CLKR is an output pin driven by the sample rate generator output clock (CLKG). |
| | DLB = 1<br>(Digital loopback mode enabled) | CLKR is an output pin driven by internal CLKX. The source for CLKX depends on the CLKXM bit. |
| CLKXM = 1 | CLKSTP = 00b or 01b<br>(Clock stop (SPI) mode disabled) | CLKX is an output pin driven by the sample rate generator output clock (CLKG). |
| | CLKSTP = 10b or 11b<br>(Clock stop (SPI) mode enabled) | The McBSP is a master in an SPI system. Internal CLKX drives internal CLKR and the shift clocks of any SPI-compliant slave devices in the system. CLKX is driven by the internal sample rate generator. |

#### 21.4.1.2 Choosing an Input Clock

The sample rate generator must be driven by an input clock signal from one of the three sources selectable with the SCLKME bit of PCR and the CLKSM bit of SRGR2 (see Table 21-5). When CLKSM = 1, the minimum divide down value in CLKGDV bits is 1. CLKGDV is described in Section 21.4.1.4.

**Table 21-5. Choosing an Input Clock for the Sample Rate Generator with the SCLKME and CLKSM Bits**

| SCLKME | CLKSM | Input Clock for Sample Rate Generator |
|---|---|---|
| 0 | 0 | Reserved |
| 0 | 1 | LSPCLK |
| 1 | 0 | Signal on MCLKR pin |
| 1 | 1 | Signal on MCLKX pin |

#### 21.4.1.3 Choosing a Polarity for the Input Clock

As shown in Figure 21-18, when the input clock is received from a pin, you can choose the polarity of the input clock. The rising edge of CLKSRG generates CLKG and FSG, but you can determine which edge of the input clock causes a rising edge on CLKSRG. The polarity options and their effects are described in Table 21-6.



Copyright © 2024 Texas Instruments Incorporated

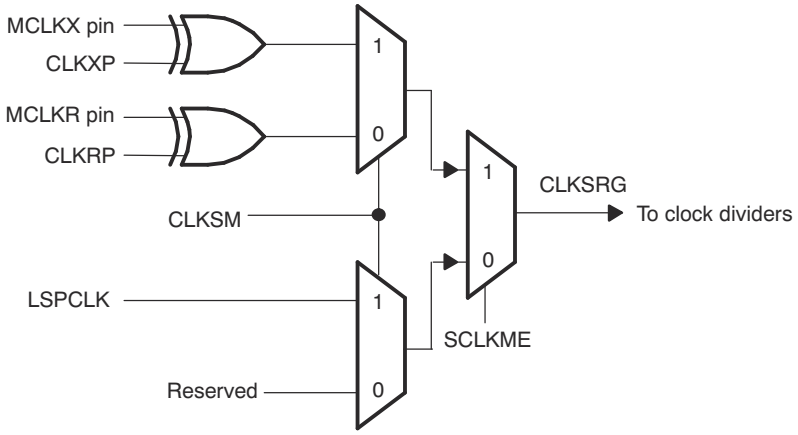



**Figure 21-18. Possible Inputs to the Sample Rate Generator and the Polarity Bits**

**Table 21-6. Polarity Options for the Input to the Sample Rate Generator**

| Input Clock | Polarity Option | Effect |
|---|---|---|
| LSPCLK | Always positive polarity | Rising edge of CPU clock generates transitions on CLKG and FSG. |
| Signal on MCLKR pin | CLKRP = 0 in PCR | Falling edge on MCLKR pin generates transitions on CLKG and FSG. |
| | CLKRP = 1 in PCR | Rising edge on MCLKR pin generates transitions on CLKG and FSG. |
| Signal on MCLKX pin | CLKXP = 0 in PCR | Rising edge on MCLKX pin generates transitions on CLKG and FSG. |
| | CLKXP = 1 in PCR | Falling edge on MCLKX pin generates transitions on CLKG and FSG. |

#### 21.4.1.4 Choosing a Frequency for the Output Clock (CLKG)

The input clock (LSPCLK or external clock) can be divided down by a programmable value to drive CLKG. Regardless of the source to the sample rate generator, the rising edge of CLKSRG (see Section 21.4.1.2) generates CLKG and FSG.

The first divider stage of the sample rate generator creates the output clock from the input clock. This divider stage uses a counter that is preloaded with the divide down value in the CLKGDV bits of SRGR1. The output of this stage is the data clock (CLKG). CLKG has the frequency represented by the following equation:

$$\text{CLKG frequency} = \frac{\text{Input clock frequency}}{(\text{CLKGDV} + 1)}$$

##### *21.4.1.4.1 CLKG Frequency*

Thus, the input clock frequency is divided by a value between 1 and 256. When CLKGDV is odd or equal to 0, the CLKG duty cycle is 50%. When CLKGDV is an even value, 2p, representing an odd divide down, the high-state duration is p+1 cycles and the low-state duration is p cycles.

#### 21.4.1.5 Keeping CLKG Synchronized to External MCLKR

When the MCLKR pin is used to drive the sample rate generator (see Section 21.4.1.2), the GSYNC bit in SRGR2 and the FSR pin can be used to configure the timing of the output clock (CLKG) relative to the input clock. Note that this feature is available only when the MCLKR pin is used to feed the external clock.

GSYNC = 1 ensures that the McBSP and an external device are dividing down the input clock with the same phase relationship. If GSYNC = 1, an inactive-to-active transition on the FSR pin triggers a resynchronization of CLKG and generation of FSG.

For more details about synchronization, see Section 21.4.3.



Copyright © 2024 Texas Instruments Incorporated

### 21.4.2 Frame Synchronization Generation in the Sample Rate Generator

The sample rate generator can produce a frame-synchronization signal (FSG) for use by the receiver, the transmitter, or both.

If you want the receiver to use FSG for frame synchronization, make sure FSRM = 1. (When FSRM = 0, receive frame synchronization is supplied via the FSR pin.)

If you want the transmitter to use FSG for frame synchronization, you must set:

- FSXM = 1 in PCR: This indicates that transmit frame synchronization is supplied by the McBSP itself rather than from the FSX pin.
- FSGM = 1 in SRGR2: This indicates that when FSXM = 1, transmit frame synchronization is supplied by the sample rate generator. (When FSGM = 0 and FSXM = 1, the transmitter uses frame-synchronization pulses generated every time data is transferred from DXR[1,2] to XSR[1,2].)

In either case, the sample rate generator must be enabled (GRST = 1) and the frame-synchronization logic in the sample rate generator must be enabled (FRST = 1).

#### 21.4.2.1 Choosing the Width of the Frame-Synchronization Pulse on FSG

Each pulse on FSG has a programmable width. You program the FWID bits of SRGR1, and the resulting pulse width is (FWID + 1) CLKG cycles, where CLKG is the output clock of the sample rate generator.

#### 21.4.2.2 Controlling the Period Between the Starting Edges of Frame-Synchronization Pulses on FSG

You can control the amount of time from the starting edge of one FSG pulse to the starting edge of the next FSG pulse. This period is controlled in one of two ways, depending on the configuration of the sample rate generator:

- If the sample rate generator is using an external input clock and GSYNC = 1 in SRGR2, FSG pulses in response to an inactive-to-active transition on the FSR pin. Thus, the frame-synchronization period is controlled by an external device.
- Otherwise, you program the FPER bits of SRGR2, and the resulting frame-synchronization period is (FPER + 1) CLKG cycles, where CLKG is the output clock of the sample rate generator.

#### 21.4.2.3 Keeping FSG Synchronized to an External Clock

When an external signal is selected to drive the sample rate generator (see Section 21.4.1.2) , the GSYNC bit of SRGR2 and the FSR pin can be used to configure the timing of FSG pulses.

GSYNC = 1 ensures that the McBSP and an external device are dividing down the input clock with the same phase relationship. If GSYNC = 1, an inactive-to-active transition on the FSR pin triggers a resynchronization of CLKG and generation of FSG.

See Section 21.4.3 for more details about synchronization.

### 21.4.3 Synchronizing Sample Rate Generator Outputs to an External Clock

The sample rate generator can produce a clock signal (CLKG) and a frame-synchronization signal (FSG) based on an input clock signal that is either the CPU clock signal or a signal at the MCLKR or MCLKX pin. When an external clock is selected to drive the sample rate generator, the GSYNC bit of SRGR2 and the FSR pin can be used to control the timing of CLKG and the pulsing of FSG relative to the chosen input clock.

Make GSYNC = 1 when you want the McBSP and an external device to divide down the input clock with the same phase relationship. If GSYNC = 1:

- An inactive-to-active transition on the FSR pin triggers a resynchronization of CLKG and a pulsing of FSG.
- CLKG always begins with a high state after synchronization.
- FSR is always detected at the same edge of the input clock signal that generates CLKG, no matter how long the FSR pulse is.
- The FPER bits of SRGR2 are ignored because the frame-synchronization period on FSG is determined by the arrival of the next frame-synchronization pulse on the FSR pin.



Copyright © 2024 Texas Instruments Incorporated

If GSYNC = 0, CLKG runs freely and is not resynchronized, and the frame-synchronization period on FSG is determined by FPER.

#### 21.4.3.1 Operating the Transmitter Synchronously with the Receiver

When GSYNC = 1, the transmitter can operate synchronously with the receiver, provided that:

- FSX is programmed to be driven by FSG (FSGM = 1 in SRGR2 and FSXM = 1 in PCR). If the input FSR has appropriate timing so that it can be sampled by the falling edge of CLKG, it can be used, instead, by setting FSXM = 0 and connecting FSR to FSX externally.
- The sample rate generator clock drives the transmit and receive clocking (CLKRM = CLKXM = 1 in PCR).

#### 21.4.3.2 Synchronization Examples

Figure 21-19 and Figure 21-20 show the clock and frame-synchronization operation with various polarities of CLKR and FSR. These figures assume FWID = 0 in SRGR1, for an FSG pulse that is one CLKG cycle wide. The FPER bits of SRGR2 are not programmed; the period from the start of a frame-synchronization pulse to the start of the next pulse is determined by the arrival of the next inactive-to-active transition on the FSR pin. Each of the figures shows what happens to CLKG when it is initially synchronized and GSYNC = 1, and when it is not initially synchronized and GSYNC = 1. Figure 21-20 has a slower CLKG frequency (it has a larger divide-down value in the CLKGDV bits of SRGR1).

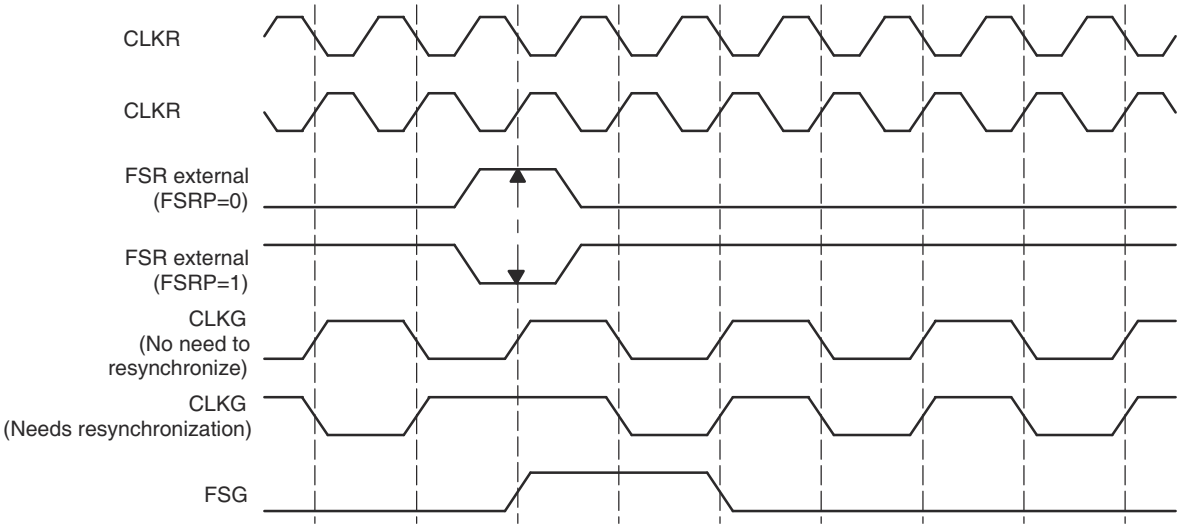



**Figure 21-19. CLKG Synchronization and FSG Generation When GSYNC = 1 and CLKGDV = 1**



Copyright © 2024 Texas Instruments Incorporated

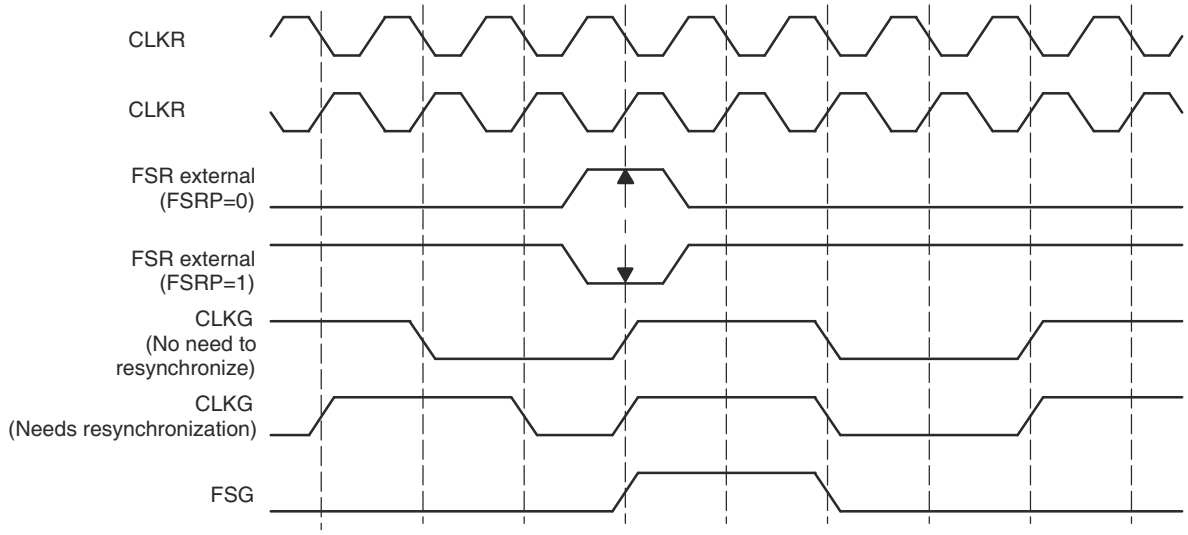



**Figure 21-20. CLKG Synchronization and FSG Generation When GSYNC = 1 and CLKGDV = 3**

### *21.4.4 Reset and Initialization Procedure for the Sample Rate Generator*

To reset and initialize the sample rate generator:

1. Place the McBSP/sample rate generator in reset.

   During a DSP reset, the sample rate generator, the receiver, and the transmitter reset bits (GRST, RRST, and XRST) are automatically forced to 0. Otherwise, during normal operation, the sample rate generator can be reset by setting GRST = 0 in SPCR2, provided that CLKG and/or FSG is not used by any portion of the McBSP. Depending on your system, you can reset the receiver (RRST = 0 in SPCR1) and reset the transmitter (XRST = 0 in SPCR2).

   If GRST = 0 due to a device reset, CLKG is driven by the CPU clock divided by 2, and FSG is driven inactive-low. If GRST = 0 due to program code, CLKG and FSG are driven low (inactive).
2. Program the registers that affect the sample rate generator.

   Program the sample rate generator registers (SRGR1 and SRGR2) as required for your application. If necessary, other control registers can be loaded with desired values, provided the respective portion of the McBSP (the receiver or transmitter) is in reset.

   After the sample rate generator registers are programmed, wait 2 CLKSRG cycles. This makes sure of proper synchronization internally.
3. Enable the sample rate generator (take the sample rate generator out of reset).

   In SPCR2, make GRST = 1 to enable the sample rate generator.

   After the sample rate generator is enabled, wait two CLKG cycles for the sample rate generator logic to stabilize.

   On the next rising edge of CLKSRG, CLKG transitions to 1 and starts clocking with a frequency equal to the following CLKG frequency equation:

$$\text{CLKG frequency} = \frac{\text{Input clock frequency}}{(\text{CLKGDV} + 1)}$$

   where the input clock is selected with the SCLKME bit of PCR and the CLKSM bit of SRGR2 in one of the configurations shown in Table 21-7.



Copyright © 2024 Texas Instruments Incorporated

**Table 21-7. Input Clock Selection for Sample Rate Generator**

| SCLKME | CLKSM | Input Clock for Sample Rate Generator |
|---|---|---|
| 0 | 0 | Reserved |
| 0 | 1 | LSPCLK |
| 1 | 0 | Signal on MCLKR pin |
| 1 | 1 | Signal on MCLKX pin |

4. If necessary, enable the receiver and/or the transmitter.

   If necessary, remove the receiver and/or transmitter from reset by setting RRST and/or XRST = 1.
5. If necessary, enable the frame-synchronization logic of the sample rate generator.

After the required data acquisition setup is done (DXR[1,2] is loaded with data), set GRST = 1 in SPCR2 if an internally generated frame-synchronization pulse is required. FSG is generated with an active-high edge after the programmed number of CLKG clocks (FPER + 1) have elapsed.

## 21.5 McBSP Exception/Error Conditions

This section describes exception/error conditions and how to handle them.

### *21.5.1 Types of Errors*

There are five serial port events that can constitute a system error:

- Receiver overrun (RFULL = 1)

  This error occurs when DRR1 has not been read since the last RBR-to-DRR copy. Consequently, the receiver does not copy a new word from the RBRs to the DRRs and the RSRs are now full with another new word shifted in from DR. Therefore, RFULL = 1 indicates an error condition wherein any new data that can arrive at this time on DR replaces the contents of the RSRs, and the previous word is lost. The RSRs continue to be overwritten as long as new data arrives on DR and DRR1 is not read. For more details about overrun in the receiver, see Section 21.5.2.
- Unexpected receive frame-synchronization pulse (RSYNCERR = 1)

  This error occurs during reception when RFIG = 0 and an unexpected frame-synchronization pulse occurs. An unexpected frame-synchronization pulse is one that begins the next frame transfer before all the bits of the current frame have been received. Such a pulse causes data reception to abort and restart. If new data has been copied into the RBRs from the RSRs since the last RBR-to-DRR copy, this new data in the RBRs is lost. This is because no RBR-to-DRR copy occurs; the reception has been restarted. For more details about receive frame-synchronization errors, see Section 21.5.3.
- Transmitter data overwrite

  This error occurs when the CPU or DMA controller overwrites data in the DXRs before the data is copied to the XSRs. The overwritten data never reaches the DX pin. For more details about overwrite in the transmitter, see Section 21.5.4.
- Transmitter underflow ($\overline{\text{XEMPTY}}$ = 0)

  If a new frame-synchronization signal arrives before new data is loaded into DXR1, the previous data in the DXRs is sent again. This procedure continues for every new frame-synchronization pulse that arrives until DXR1 is loaded with new data. For more details about underflow in the transmitter, see Section 21.5.5.
- Unexpected transmit frame-synchronization pulse (XSYNCERR = 1)

  This error occurs during transmission when XFIG = 0 and an unexpected frame-synchronization pulse occurs. An unexpected frame-synchronization pulse is one that begins the next frame transfer before all the bits of the current frame have been transferred. Such a pulse causes the current data transmission to abort and restart. If new data has been written to the DXRs since the last DXR-to-XSR copy, the current value in the XSRs is lost. For more details about transmit frame-synchronization errors, see Section 21.5.6.



Copyright © 2024 Texas Instruments Incorporated

### 21.5.2 Overrun in the Receiver

RFULL = 1 in SPCR1 indicates that the receiver has experienced overrun and is in an error condition. RFULL is set when all of the following conditions are met:

1. DRR1 has not been read since the last RBR-to-DRR copy (RRDY = 1).
2. RBR1 is full and an RBR-to-DRR copy has not occurred.
3. RSR1 is full and an RSR1-to-RBR copy has not occurred.

As described in Section 21.3.5, data arriving on DR is continuously shifted into RSR1 (for word length of 16 bits or smaller) or RSR2 and RSR1 (for word length larger than 16 bits). Once a complete word is shifted into the RSR(s), an RSR-to-RBR copy can occur only if the previous data in RBR1 has been copied to DRR1. The RRDY bit is set when new data arrives in DRR1 and is cleared when that data is read from DRR1. Until RRDY = 0, the next RBR-to-DRR copy does not take place, and the data is held in the RSR(s). New data arriving on the DR pin is shifted into RSR(s), and the previous content of the RSR(s) is lost.

You can prevent the loss of data if DRR1 is read no later than 2.5 cycles before the end of the third word is shifted into the RSR1.

**Note**

If both DRRs are needed (word length larger than 16 bits), the CPU or the DMA controller must read from DRR2 first and then from DRR1. As soon as DRR1 is read, the next RBR-to-DRR copy is initiated. If DRR2 is not read first, the data in DRR2 is lost.

After the receiver starts running from reset, a minimum of three words must be received before RFULL is set. Either of the following events clears the RFULL bit and allows subsequent transfers to be read properly:

- The CPU or DMA controller reads DRR1.
- The receiver is reset individually (RRST = 0) or as part of a device reset.

Another frame-synchronization pulse is required to restart the receiver.

#### 21.5.2.1 Example of Overrun Condition

Figure 21-21 shows the receive overrun condition. Because serial word A is not read from DRR1 before serial word B arrives in RBR1, B is not transferred to DRR1 yet. Another new word ©) arrives and RSR1 is full with this data. DRR1 is finally read, but not earlier than 2.5 cycles before the end of word C. Therefore, new data (D) overwrites word C in RSR1. If DRR1 is not read in time, the next word can overwrite D.

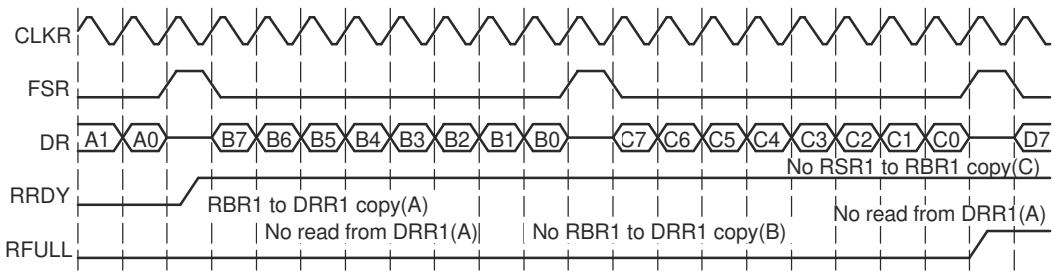



**Figure 21-21. Overrun in the McBSP Receiver**



Copyright © 2024 Texas Instruments Incorporated

#### 21.5.2.2 Example of Preventing Overrun Condition

Figure 21-22 shows the case where RFULL is set, but the overrun condition is prevented by a read from DRR1 at least 2.5 cycles before the next serial word is completely shifted into RSR1. This makes sure that an RBR1-to-DRR1 copy of word B occurs before receiver attempts to transfer word C from RSR1 to RBR1.

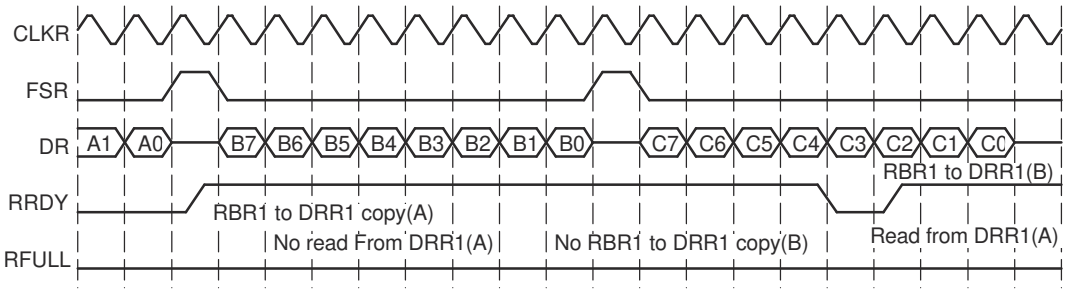



**Figure 21-22. Overrun Prevented in the McBSP Receiver**

### *21.5.3 Unexpected Receive Frame-Synchronization Pulse*

Section 21.5.3.1 shows how the McBSP responds to any receive frame-synchronization pulses, including an unexpected pulse. Section 21.5.3.2 and Section 21.5.3.3 show an example of a frame-synchronization error and an example of how to prevent such an error, respectively.

#### 21.5.3.1 Possible Responses to Receive Frame-Synchronization Pulses

Figure 21-23 shows the decision tree that the receiver uses to handle all incoming frame-synchronization pulses. The figure assumes that the receiver has been started (RRST = 1 in SPCR1). Case 3 shows where an error occurs.

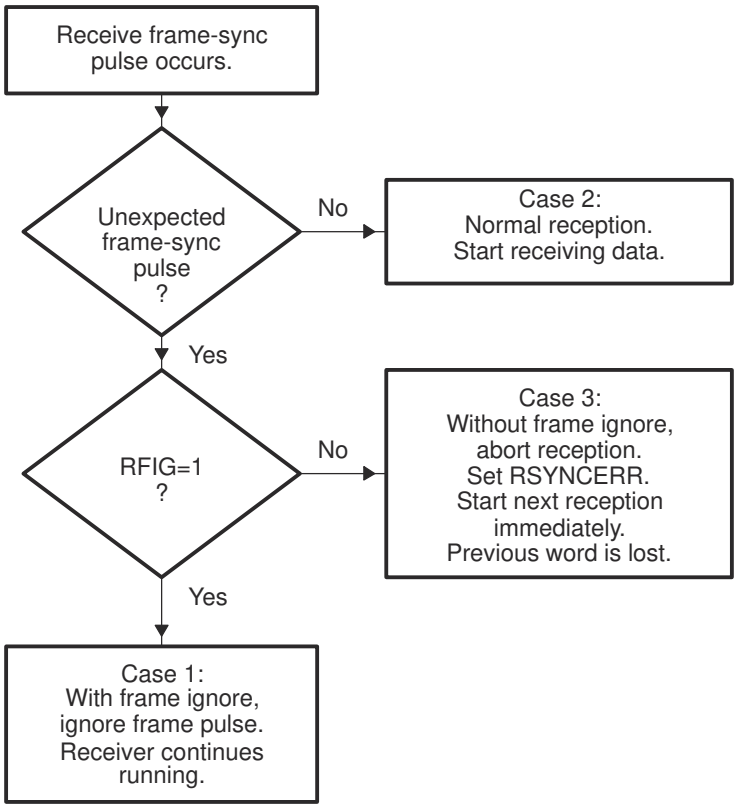



**Figure 21-23. Possible Responses to Receive Frame-Synchronization Pulses**



Copyright © 2024 Texas Instruments Incorporated

Any one of three cases can occur:

- Case 1: Unexpected internal FSR pulses with RFIG = 1 in RCR2. Receive frame-synchronization pulses are ignored, and the reception continues.
- Case 2: Normal serial port reception. Reception continues normally because the frame-synchronization pulse is not unexpected. There are three possible reasons why a receive operation might *not* be in progress when the pulse occurs:
  - The FSR pulse is the first after the receiver is enabled (RRST = 1 in SPCR1).
  - The FSR pulse is the first after DRR[1,2] is read, clearing a receiver full (RFULL = 1 in SPCR1) condition.
  - The serial port is in the interpacket intervals. The programmed data delay for reception (programmed with the RDATDLY bits in RCR2) may start during these interpacket intervals for the first bit of the next word to be received. Thus, at maximum frame frequency, frame synchronization can still be received 0 to 2 clock cycles before the first bit of the synchronized frame.
- Case 3: Unexpected receive frame synchronization with RFIG = 0 (frame-synchronization pulses not ignored). Unexpected frame-synchronization pulses can originate from an external source or from the internal sample rate generator.

  If a frame-synchronization pulse starts the transfer of a new frame before the current frame is fully received, this pulse is treated as an unexpected frame-synchronization pulse, and the receiver sets the receive frame-synchronization error bit (RSYNCERR) in SPCR1. RSYNCERR can be cleared only by a receiver reset or by a write of 0 to this bit.

  If you want the McBSP to notify the CPU of receive frame-synchronization errors, you can set a special receive interrupt mode with the RINTM bits of SPCR1. When RINTM = 11b, the McBSP sends a receive interrupt (RINT) request to the CPU each time that RSYNCERR is set.

#### 21.5.3.2 Example of Unexpected Receive Frame-Synchronization Pulse

Figure 21-30 shows an unexpected receive frame-synchronization pulse during normal operation of the serial port, with time intervals between data packets. When the unexpected frame-synchronization pulse occurs, the RSYNCERR bit is set, the reception of data B is aborted, and the reception of data C begins. In addition, if RINTM = 11b, the McBSP sends a receive interrupt (RINT) request to the CPU.

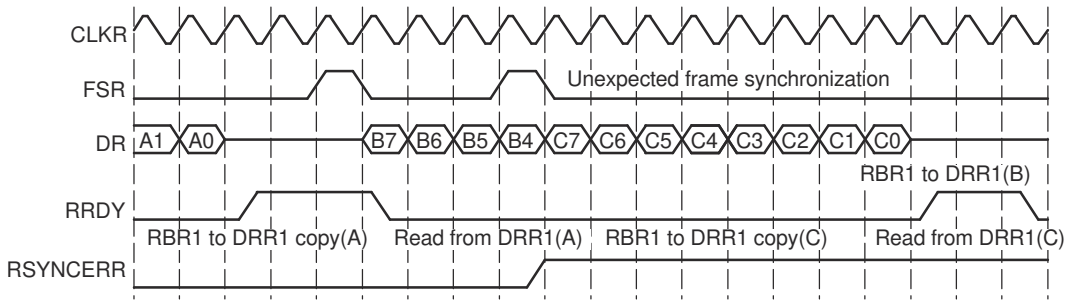



**Figure 21-24. An Unexpected Frame-Synchronization Pulse During a McBSP Reception**

#### 21.5.3.3 Preventing Unexpected Receive Frame-Synchronization Pulses

Each frame transfer can be delayed by 0, 1, or 2 MCLKR cycles, depending on the value in the RDATDLY bits of RCR2. For each possible data delay, Figure 21-25 shows when a new frame-synchronization pulse on FSR can safely occur relative to the last bit of the current frame.



Copyright © 2024 Texas Instruments Incorporated

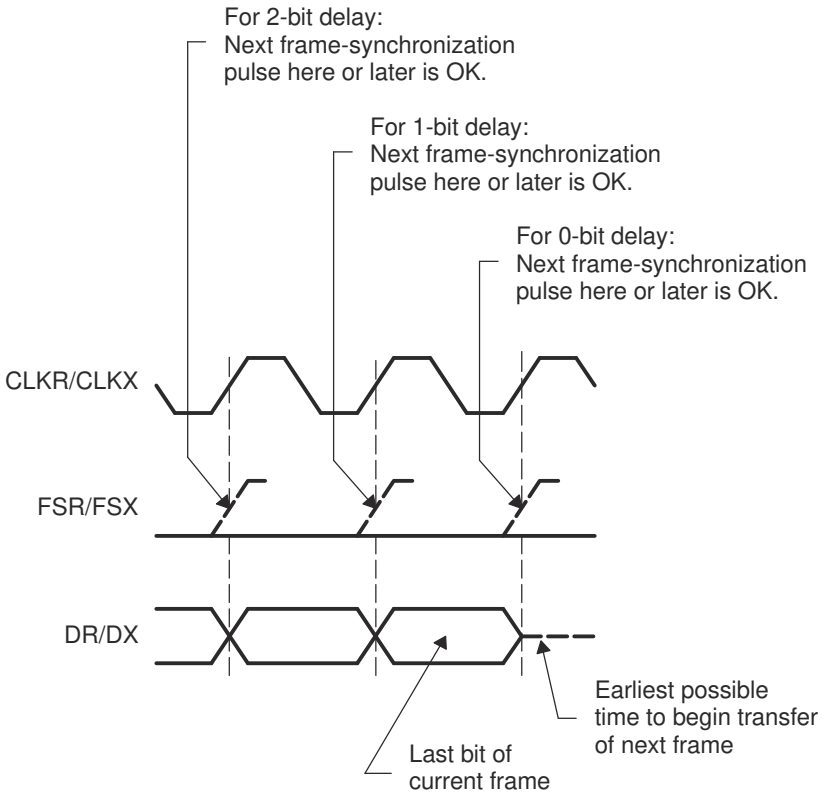



**Figure 21-25. Proper Positioning of Frame-Synchronization Pulses**

### *21.5.4 Overwrite in the Transmitter*

As described in Section 21.3.6), the transmitter must copy the data previously written to the DXR(s) by the CPU or DMA controller into the XSR(s) and then shift each bit from the XSR(s) to the DX pin. If new data is written to the DXR(s) before the previous data is copied to the XSR(s), the previous data in the DXR(s) is overwritten and thus lost.

#### 21.5.4.1 Example of Overwrite Condition

Figure 21-26 shows what happens if the data in DXR1 is overwritten before being transmitted. Initially, DXR1 is loaded with data C. A subsequent write to DXR1 overwrites C with D before C is copied to XSR1. Thus, C is never transmitted on DX.

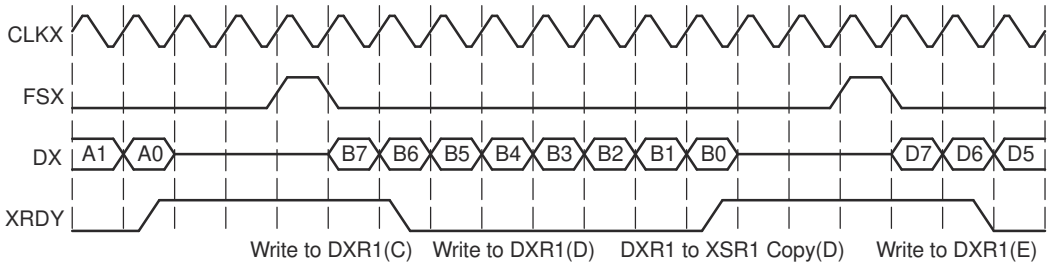



**Figure 21-26. Data in the McBSP Transmitter Overwritten and Thus Not Transmitted**

#### 21.5.4.2 Preventing Overwrites

You can prevent CPU overwrites by making the CPU:

- Poll for XRDY = 1 in SPCR2 before writing to the DXRs. XRDY is set when data is copied from DXR1 to XSR1 and is cleared when new data is written to DXR1.



Copyright © 2024 Texas Instruments Incorporated

- Wait for a transmit interrupt (XINT) before writing to the DXRs. When XINTM = 00b in SPCR2, the transmitter sends XINT to the CPU each time XRDY is set.

You can prevent DMA overwrites by synchronizing DMA transfers to the transmit synchronization event XEVT. The transmitter sends an XEVT signal each time XRDY is set.

### *21.5.5 Underflow in the Transmitter*

The McBSP indicates a transmitter empty (or underflow) condition by clearing the $\overline{\text{XEMPTY}}$ bit in SPCR2. Either of the following events activates $\overline{\text{XEMPTY}}$ ( $\overline{\text{XEMPTY}}$ = 0):

- DXR1 has not been loaded since the last DXR-to-XSR copy, and all bits of the data word in the XSR(s) have been shifted out on the DX pin.
- The transmitter is reset (by forcing XRST = 0 in SPCR2, or by a device reset) and is then restarted.

In the underflow condition, the transmitter continues to transmit the old data that is in the DXR(s) for every new transmit frame-synchronization signal, until a new value is loaded into DXR1 by the CPU or the DMA controller.

**Note**

If both DXRs are needed (word length larger than 16 bits), the CPU or the DMA controller must load DXR2 first and then load DXR1. As soon as DXR1 is loaded, the contents of both DXRs are copied to the transmit shift registers (XSRs). If DXR2 is not loaded first, the previous content of DXR2 is passed to the XSR2.

XEMPTY is deactivated (XEMPTY = 1) when a new word in DXR1 is transferred to XSR1. If FSXM = 1 in PCR and FSGM = 0 in SRGR2, the transmitter generates a single internal FSX pulse in response to a DXR-to-XSR copy. Otherwise, the transmitter waits for the next frame-synchronization pulse before sending out the next frame on DX.

When the transmitter is taken out of reset (XRST = 1), it is in a transmitter ready (XRDY = 1 in SPCR2) and transmitter empty (XEMPTY = 0) state. If DXR1 is loaded by the CPU or the DMA controller before internal FSX goes active high, a valid DXR-to-XSR transfer occurs. This allows for the first word of the first frame to be valid even before the transmit frame-synchronization pulse is generated or detected. Alternatively, if a transmit frame-synchronization pulse is detected before DXR1 is loaded, zeros are output on DX.

#### 21.5.5.1 Example of the Underflow Condition

Figure 21-27 shows an underflow condition. After B is transmitted, DXR1 is not reloaded before the subsequent frame-synchronization pulse. Thus, B is again transmitted on DX.

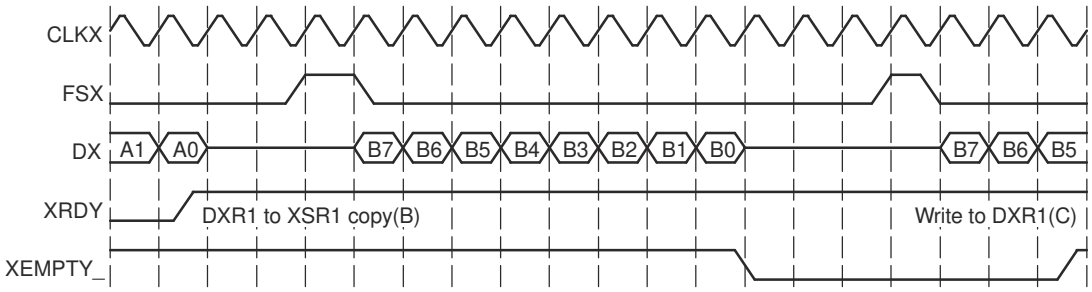



**Figure 21-27. Underflow During McBSP Transmission**

#### 21.5.5.2 Example of Preventing Underflow Condition

Figure 21-28 shows the case of writing to DXR1 just before an underflow condition can otherwise occur. After B is transmitted, C is written to DXR1 before the next frame-synchronization pulse. As a result, there is no underflow; B is not transmitted twice.



Copyright © 2024 Texas Instruments Incorporated

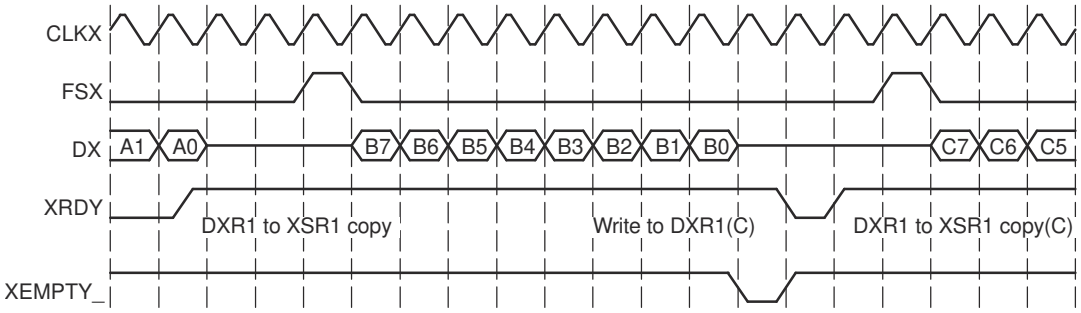



**Figure 21-28. Underflow Prevented in the McBSP Transmitter**

### *21.5.6 Unexpected Transmit Frame-Synchronization Pulse*

Section 21.5.6.1 shows how the McBSP responds to any transmit frame-synchronization pulses, including an unexpected pulse. Section 21.5.6.2 and Section 21.5.6.3 show examples of a frame-synchronization error and an example of how to prevent such an error, respectively.

#### 21.5.6.1 Possible Responses to Transmit Frame-Synchronization Pulses

Figure 21-29 shows the decision tree that the transmitter uses to handle all incoming frame-synchronization pulses. The figure assumes that the transmitter has been started (XRST = 1 in SPCR2). Case 3 shows where an error occurs.

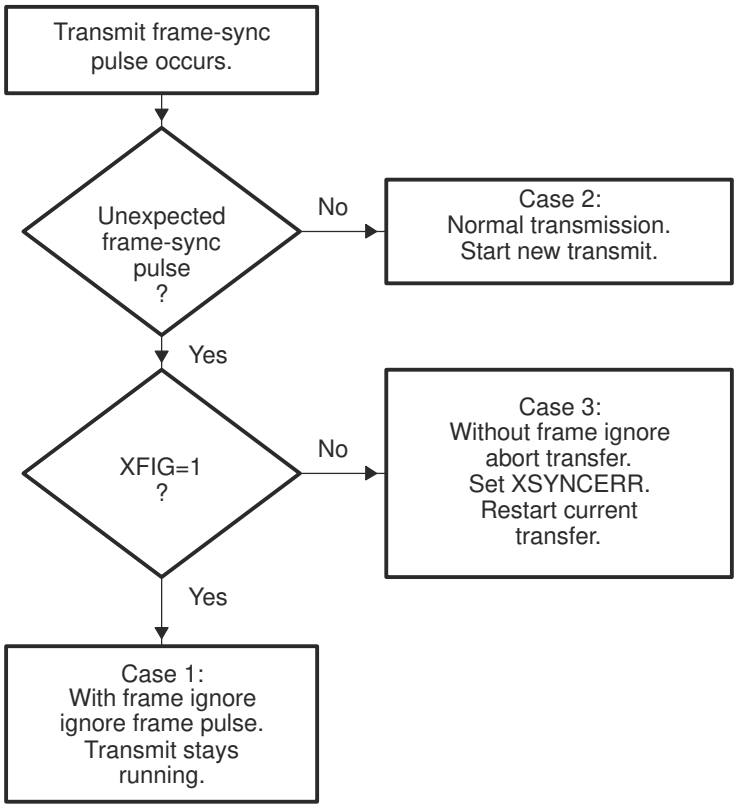



**Figure 21-29. Possible Responses to Transmit Frame-Synchronization Pulses**



Copyright © 2024 Texas Instruments Incorporated

Any one of three cases can occur:

- Case 1: Unexpected internal FSX pulses with XFIG = 1 in XCR2. Transmit frame-synchronization pulses are ignored, and the transmission continues.
- Case 2: Normal serial port transmission. Transmission continues normally because the frame-synchronization pulse is not unexpected. There are two possible reasons why a transmit operations might *not* be in progress when the pulse occurs:

  This FSX pulse is the first after the transmitter is enabled (XRST = 1).

  The serial port is in the interpacket intervals. The programmed data delay for transmission (programmed with the XDATDLY bits of XCR2) may start during these interpacket intervals before the first bit of the previous word is transmitted. Thus, at maximum packet frequency, frame synchronization can still be received 0 to 2 clock cycles before the first bit of the synchronized frame.
- Case 3: Unexpected transmit frame synchronization with XFIG = 0 (frame-synchronization pulses not ignored). Unexpected frame-synchronization pulses can originate from an external source or from the internal sample rate generator.

  If a frame-synchronization pulse starts the transfer of a new frame before the current frame is fully transmitted, this pulse is treated as an unexpected frame-synchronization pulse, and the transmitter sets the transmit frame-synchronization error bit (XSYNCERR) in SPCR2. XSYNCERR can be cleared only by a transmitter reset or by a write of 0 to this bit.

  If you want the McBSP to notify the CPU of frame-synchronization errors, you can set a special transmit interrupt mode with the XINTM bits of SPCR2. When XINTM = 11b, the McBSP sends a transmit interrupt (XINT) request to the CPU each time that XSYNCERR is set.

#### 21.5.6.2 Example of Unexpected Transmit Frame-Synchronization Pulse

Section 21.5.3.2 shows an unexpected transmit frame-synchronization pulse during normal operation of the serial port with intervals between the data packets. When the unexpected frame-synchronization pulse occurs, the XSYNCERR bit is set and the transmission of data B is restarted because no new data has been passed to XSR1 yet. In addition, if XINTM = 11b, the McBSP sends a transmit interrupt (XINT) request to the CPU.

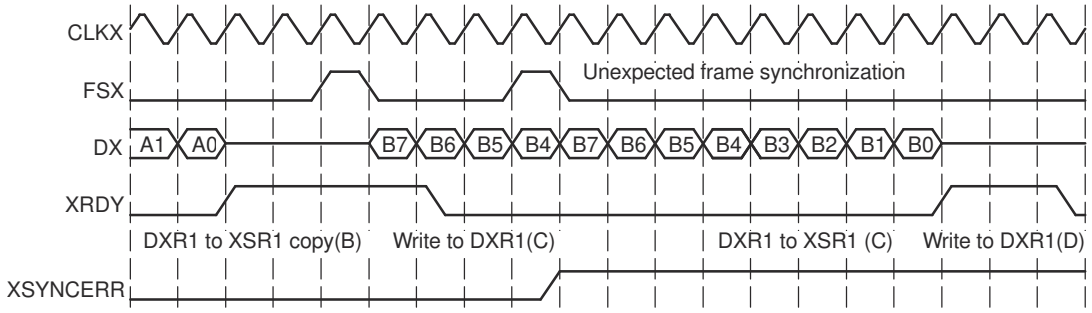



**Figure 21-30. An Unexpected Frame-Synchronization Pulse During a McBSP Transmission**



Copyright © 2024 Texas Instruments Incorporated

#### 21.5.6.3 Preventing Unexpected Transmit Frame-Synchronization Pulses

Each frame transfer can be delayed by 0, 1, or 2 CLKX cycles, depending on the value in the XDATDLY bits of XCR2. For each possible data delay, Figure 21-31 shows when a new frame-synchronization pulse on FSX can safely occur relative to the last bit of the current frame.

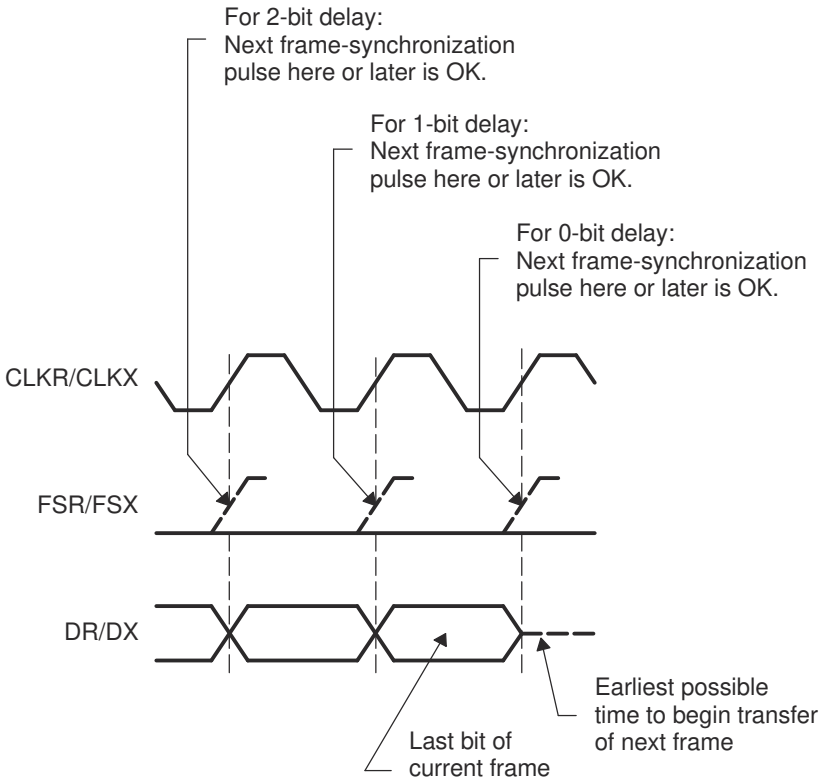



**Figure 21-31. Proper Positioning of Frame-Synchronization Pulses**



Copyright © 2024 Texas Instruments Incorporated

## 21.6 Multichannel Selection Modes

This section discusses the multichannel selection modes for the McBSP.

### *21.6.1 Channels, Blocks, and Partitions*

A McBSP channel is a time slot for shifting in/out the bits of one serial word. Each McBSP supports up to 128 channels for reception and 128 channels for transmission.

In the receiver and in the transmitter, the 128 available channels are divided into eight blocks that each contain 16 contiguous channels (see Table 21-8 through Table 21-10):

- It is possible to have two receive partitions (A and B) and 8 transmit partitions (A to H).
- McBSP can transmit/receive on selected channels.
- Each channel partition has a dedicated channel-enable register. Each bit controls whether data flow is allowed or prevented in one of the channels assigned to that partition.
- There are three transmit multichannel modes and one receive multichannel mode.

The blocks are assigned to partitions according to the selected partition mode. In the two-partition mode (described in Section 21.6.4), you assign one even-numbered block (0, 2, 4, or 6) to partition A and one odd-numbered block (1, 3, 5, or 7) to partition B. In the 8-partition mode (described in Section 21.6.5), blocks 0 through 7 are automatically assigned to partitions, A through H, respectively.

The number of partitions for reception and the number of partitions for transmission are independent. For example, using two receive partitions (A and B) and eight transmit partitions (A to H) is possible.

**Table 21-8. Block - Channel Assignment**

| Block | Channels |
|---|---|
| 0 | 0 -15 |
| 1 | 16 - 31 |
| 2 | 32 - 47 |
| 3 | 48 - 63 |
| 4 | 64 - 79 |
| 5 | 80 - 95 |
| 6 | 96 - 111 |
| 7 | 112 - 127 |

**Table 21-9. 2-Partition Mode**

| Partition | Blocks |
|---|---|
| A | 0, 2, 4, or 6 |
| B | 1, 3, 5, or 7 |

**Table 21-10. 8-Partition Mode**

| Partition | Blocks | Channels |
|---|---|---|
| A | 0 | 0 -15 |
| B | 1 | 16 - 31 |
| C | 2 | 32 - 47 |
| D | 3 | 48 - 63 |
| E | 4 | 64 - 79 |
| F | 5 | 80 - 95 |
| G | 6 | 96 - 111 |
| H | 7 | 112 - 127 |



Copyright © 2024 Texas Instruments Incorporated

### 21.6.2 Multichannel Selection

When a McBSP uses a time-division multiplexed (TDM) data stream while communicating with other McBSPs or serial devices, the McBSP may need to receive and/or transmit on only a few channels. To save memory and bus bandwidth, you can use a multichannel selection mode to prevent data flow in some of the channels.

Each channel partition has a dedicated channel enable register. If the appropriate multichannel selection mode is on, each bit in the register controls whether data flow is allowed or prevented in one of the channels that is assigned to that partition.

The McBSP has one receive multichannel selection mode (described in Section 21.6.6) and three transmit multichannel selection modes (described in Section 21.6.7).

### 21.6.3 Configuring a Frame for Multichannel Selection

Before you enable a multichannel selection mode, make sure you properly configure the data frame:

- Select a single-phase frame (RPHASE/XPHASE = 0). Each frame represents a TDM data stream.
- Set a frame length (in RFRLEN1/XFRLEN1) that includes the highest-numbered channel to be used. For example, if you plan to use channels 0, 15, and 39 for reception, the receive frame length must be at least 40 (RFRLEN1 = 39). If XFRLEN1 = 39 in this case, the receiver creates 40 time slots per frame but only receives data during time slots 0, 15, and 39 of each frame.

### 21.6.4 Using Two Partitions

For multichannel selection operation in the receiver and/or the transmitter, you can use two partitions or eight partitions (described in Section 21.6.5). If you choose the 2-partition mode (RMCME = 0 for reception, XMCME = 0 for transmission), McBSP channels are activated using an alternating scheme. In response to a frame-synchronization pulse, the receiver or transmitter begins with the channels in partition A and then alternates between partitions B and A until the complete frame has been transferred. When the next frame-synchronization pulse occurs, the next frame is transferred beginning with the channels in partition A.

#### 21.6.4.1 Assigning Blocks to Partitions A and B

For reception, any two of the eight receive-channel blocks can be assigned to receive partitions A and B, which means up to 32 receive channels can be enabled at any given point in time. Similarly, any two of the eight transmit-channel blocks (up 32 enabled transmit channels) can be assigned to transmit partitions A and B.

For reception:

- Assign an even-numbered channel block (0, 2, 4, or 6) to receive partition A by writing to the RPABLK bits. In the receive multichannel selection mode (described in Section 21.6.6), the channels in this partition are controlled by receive channel enable register A (RCERA).
- Assign an odd-numbered block (1, 3, 5, or 7) to receive partition B with the RPBBLK bits. In the receive multichannel selection mode, the channels in this partition are controlled by receive channel enable register B (RCERB).

For transmission:

- Assign an even-numbered channel block (0, 2, 4, or 6) to transmit partition A by writing to the XPABLK bits. In one of the transmit multichannel selection modes (described in Section 21.6.7), the channels in this partition are controlled by transmit channel enable register A (XCERA).
- Assign an odd-numbered block (1, 3, 5, or 7) to transmit partition B with the XPBBLK bits. In one of the transmit multichannel selection modes, the channels in this partition are controlled by transmit channel enable register B (XCERB).

Figure 21-32 shows an example of alternating between the channels of partition A and the channels of partition B. Channels 0-15 have been assigned to partition A, and channels 16-31 have been assigned to partition B. In response to a frame-synchronization pulse, the McBSP begins a frame transfer with partition A and then alternates between partitions B and A until the complete frame is transferred.



Copyright © 2024 Texas Instruments Incorporated

As explained in Section 21.6.4.2, you can dynamically change which blocks of channels are assigned to the partitions.

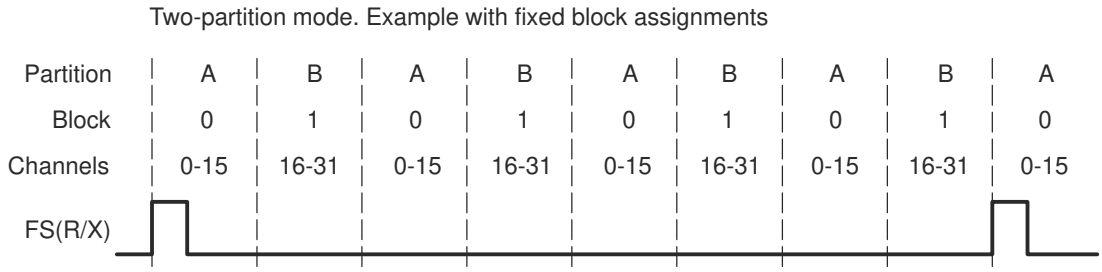



**Figure 21-32. Alternating Between the Channels of Partition A and the Channels of Partition B**

#### 21.6.4.2 Reassigning Blocks During Reception/Transmission

If you want to use more than 32 channels, you can change which channel blocks are assigned to partitions A and B during the course of a data transfer. However, these changes must be carefully timed. While a partition is being transferred, the associated block assignment bits cannot be modified and the associated channel enable register cannot be modified. For example, if block 3 is being transferred and block 3 is assigned to partition A, you cannot modify (R/X)PABLK to assign different channels to partition A, nor (R/X)CERA to change the channel configuration for partition A.

Several features of the McBSP help you time the reassignment:

- The block of channels currently involved in reception and transmission (the current block) is reflected in the RCBLK/XCBLK bits. Your program can poll these bits to determine which partition is active. When a partition is not active, it is safe to change its block assignment and channel configuration.
- At the end of every block (at the boundary of two partitions), an interrupt can be sent to the CPU. In response to the interrupt, the CPU can then check the RCBLK/XCBLK bits and update the inactive partition. See Section 21.6.8.

Figure 21-33 shows an example of reassigning channels throughout a data transfer. In response to a frame-synchronization pulse, the McBSP alternates between partitions A and B. Whenever partition B is active, the CPU changes the block assignment for partition A. Whenever partition A is active, the CPU changes the block assignment for partition B.

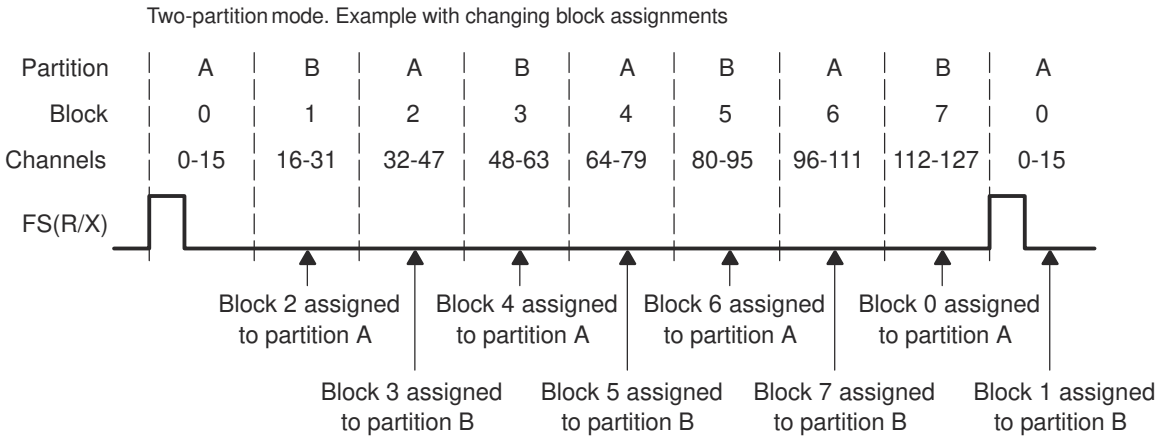



**Figure 21-33. Reassigning Channel Blocks Throughout a McBSP Data Transfer**



Copyright © 2024 Texas Instruments Incorporated

### 21.6.5 Using Eight Partitions

For multichannel selection operation in the receiver and/or the transmitter, you can use eight partitions or two partitions (described in Section 21.6.4). If you choose the 8-partition mode (RMCME = 1 for reception, XMCME = 1 for transmission), McBSP channels are activated in the following order: A, B, C, D, E, F, G, H. In response to a frame-synchronization pulse, the receiver or transmitter begins with the channels in partition A and then continues with the other partitions in order until the complete frame has been transferred. When the next frame-synchronization pulse occurs, the next frame is transferred, beginning with the channels in partition A.

In the 8-partition mode, the (R/X)PABLK and (R/X)PBBLK bits are ignored and the 16-channel blocks are assigned to the partitions as shown in Table 21-11 and Table 21-12. These assignments cannot be changed. The tables also show the registers used to control the channels in the partitions.

**Table 21-11. Receive Channel Assignment and Control With Eight Receive Partitions**

| Receive Partition | Assigned Block of Receive Channels | Register Used For Channel Control |
|---|---|---|
| A | Block 0: channels 0 through 15 | RCERA |
| B | Block 1: channels 16 through 31 | RCERB |
| C | Block 2: channels 32 through 47 | RCERC |
| D | Block 3: channels 48 through 63 | RCERD |
| E | Block 4: channels 64 through 79 | RCERE |
| F | Block 5: channels 80 through 95 | RCERF |
| G | Block 6: channels 96 through 111 | RCERG |
| H | Block 7: channels 112 through 127 | RCERH |

**Table 21-12. Transmit Channel Assignment and Control When Eight Transmit Partitions Are Used**

| Transmit Partition | Assigned Block of Transmit Channels | Register Used For Channel Control |
|---|---|---|
| A | Block 0: channels 0 through 15 | XCERA |
| B | Block 1: channels 16 through 31 | XCERB |
| C | Block 2: channels 32 through 47 | XCERC |
| D | Block 3: channels 48 through 63 | XCERD |
| E | Block 4: channels 64 through 79 | XCERE |
| F | Block 5: channels 80 through 95 | XCERF |
| G | Block 6: channels 96 through 111 | XCERG |
| H | Block 7: channels 112 through 127 | XCERH |

Figure 21-34 shows an example of the McBSP using the 8-partition mode. In response to a frame-synchronization pulse, the McBSP begins a frame transfer with partition A and then activates B, C, D, E, F, G, and H to complete a 128-word frame.

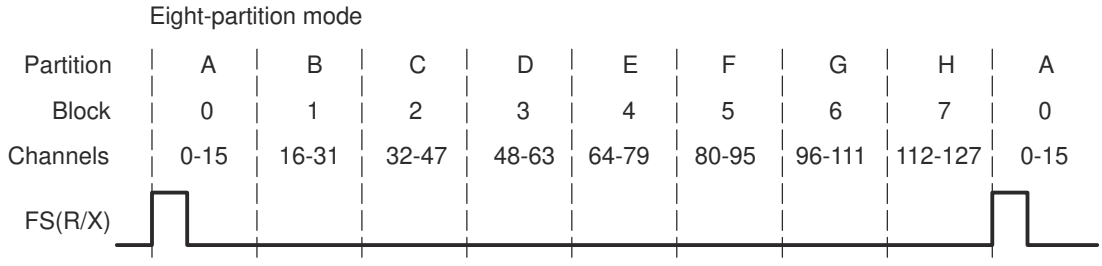



**Figure 21-34. McBSP Data Transfer in the 8-Partition Mode**



Copyright © 2024 Texas Instruments Incorporated

### 21.6.6 Receive Multichannel Selection Mode

The RMCM bit of MCR1 determines whether all channels or only selected channels are enabled for reception. When RMCM = 0, all 128 receive channels are enabled and cannot be disabled. When RMCM = 1, the receive multichannel selection mode is enabled. In this mode:

- Channels can be individually enabled or disabled. The only channels enabled are those selected in the appropriate receive channel enable registers (RCERs). The way channels are assigned to the RCERs depends on the number of receive channel partitions (2 or 8), as defined by the RMCME bit of MCR1.
- If a receive channel is disabled, any bits received in that channel are passed only as far as the receive buffer registers (RBRs). The receiver does not copy the content of the RBRs to the DRRs, and as a result, does not set the receiver ready bit (RRDY). Therefore, no DMA synchronization event (REVT) is generated and, if the receiver interrupt mode depends on RRDY (RINTM = 00b), no interrupt is generated.

As an example of how the McBSP behaves in the receive multichannel selection mode, suppose you enable only channels 0, 15, and 39 and that the frame length is 40. The McBSP:

1. Accepts bits shifted in from the DR pin in channel 0
2. Ignores bits received in channels 1-14
3. Accepts bits shifted in from the DR pin in channel 15
4. Ignores bits received in channels 16-38
5. Accepts bits shifted in from the DR pin in channel 39



Copyright © 2024 Texas Instruments Incorporated

### 21.6.7 Transmit Multichannel Selection Modes

The XMCM bits of XCR2 determine whether all channels or only selected channels are enabled and unmasked for transmission. More details on enabling and masking are in Section 21.6.7.1. The McBSP has three transmit multichannel selection modes (XMCM = 01b, XMCM = 10b, and XMCM = 11b), which are described in Table 21-13.

**Table 21-13. Selecting a Transmit Multichannel Selection Mode With the XMCM Bits**

| XMCM | Transmit Multichannel Selection Mode |
|---|---|
| 00b | No transmit multichannel selection mode is on. All channels are enabled and unmasked. No channels can be disabled or masked. |
| 01b | All channels are disabled unless the channels are selected in the appropriate transmit channel enable registers (XCERs). If enabled, a channel in this mode is also unmasked.<br>The XMCME bit of MCR2 determines whether 32 channels or 128 channels are selectable in XCERs. |
| 10b | All channels are enabled, but the channels are masked unless the channels are selected in the appropriate transmit channel enable registers (XCERs).<br>The XMCME bit of MCR2 determines whether 32 channels or 128 channels are selectable in XCERs. |
| 11b | This mode is used for symmetric transmission and reception.<br>All channels are disabled for transmission unless the channels are enabled for reception in the appropriate receive channel enable registers (RCERs). Once enabled, the channels are masked unless the channels are also selected in the appropriate transmit channel enable registers (XCERs).<br>The XMCME bit of MCR2 determines whether 32 channels or 128 channels are selectable in RCERs and XCERs. |

As an example of how the McBSP behaves in a transmit multichannel selection mode, suppose that XMCM = 01b (all channels disabled unless individually enabled) and that you have enabled only channels 0, 15, and 39. Suppose also that the frame length is 40. The McBSP:

1. Shifts data to the DX pin in channel 0
2. Places the DX pin in the high impedance state in channels 1-14
3. Shifts data to the DX pin in channel 15
4. Places the DX pin in the high impedance state in channels 16-38
5. Shifts data to the DX pin in channel 39

#### 21.6.7.1 Disabling/Enabling Versus Masking/Unmasking

For transmission, a channel can be:

- Enabled and unmasked (transmission can begin and can be completed)
- Enabled but masked (transmission can begin but cannot be completed)
- Disabled (transmission cannot occur)

The following definitions explain the channel control options:

**Enabled channel** — A channel that can begin transmission by passing data from the data transmit registers (DXRs) to the transmit shift registers (XSRs).

**Masked channel** — A channel that cannot complete transmission. The DX pin is held in the high impedance state; data cannot be shifted out on the DX pin.

In systems where symmetric transmit and receive provides software benefits, this feature allows transmit channels to be disabled on a shared serial bus. A similar feature is not needed for reception because multiple receptions cannot cause serial bus contention.



Copyright © 2024 Texas Instruments Incorporated

| | |
|---|---|
| **Disabled channel** | A channel that is not enabled. A disabled channel is also masked.<br>Because no DXR-to-XSR copy occurs, the XRDY bit of SPCR2 is not set. Therefore, no DMA synchronization event (XEVT) is generated, and if the transmit interrupt mode depends on XRDY (XINTM = 00b in SPCR2), no interrupt is generated.<br>The XEMPTY bit of SPCR2 is not affected. |
| **Unmasked channel** | A channel that is not masked. Data in the XSRs is shifted out on the DX pin. |

#### 21.6.7.2 Activity on McBSP Pins for Different Values of XMCM

Figure 21-35 shows the activity on the McBSP pins for the various XMCM values. In all cases, the transmit frame is configured as follows:

- XPHASE = 0: Single-phase frame (required for multichannel selection modes)
- XFRLEN1 = 0000011b: 4 words per frame
- XWDLEN1 = 000b: 8 bits per word
- XMCME = 0: 2-partition mode (only partitions A and B used)

In the case where XMCM = 11b, transmission and reception are symmetric, which means the corresponding bits for the receiver (RPHASE, RFRLEN1, RWDLEN1, and RMCME) must have the same values as XPHASE, XFRLEN1, and XWDLEN1, respectively.

In the figure, the arrows showing where the various events occur are only sample indications. Wherever possible, there is a time window in which these events can occur.

### *21.6.8 Using Interrupts Between Block Transfers*

When a multichannel selection mode is used, an interrupt request can be sent to the CPU at the end of every 16-channel block (at the boundary between partitions and at the end of the frame). In the receive multichannel selection mode, a receive interrupt (RINT) request is generated at the end of each block transfer if RINTM = 01b. In any of the transmit multichannel selection modes, a transmit interrupt (XINT) request is generated at the end of each block transfer if XINTM = 01b. When RINTM/XINTM = 01b, no interrupt is generated unless a multichannel selection mode is on.

These interrupt pulses are active high and last for two CPU clock cycles.

This type of interrupt is especially helpful if you are using the two-partition mode (described in Section 21.6.4) and you want to know when you can assign a different block of channels to partition A or B.



Copyright © 2024 Texas Instruments Incorporated

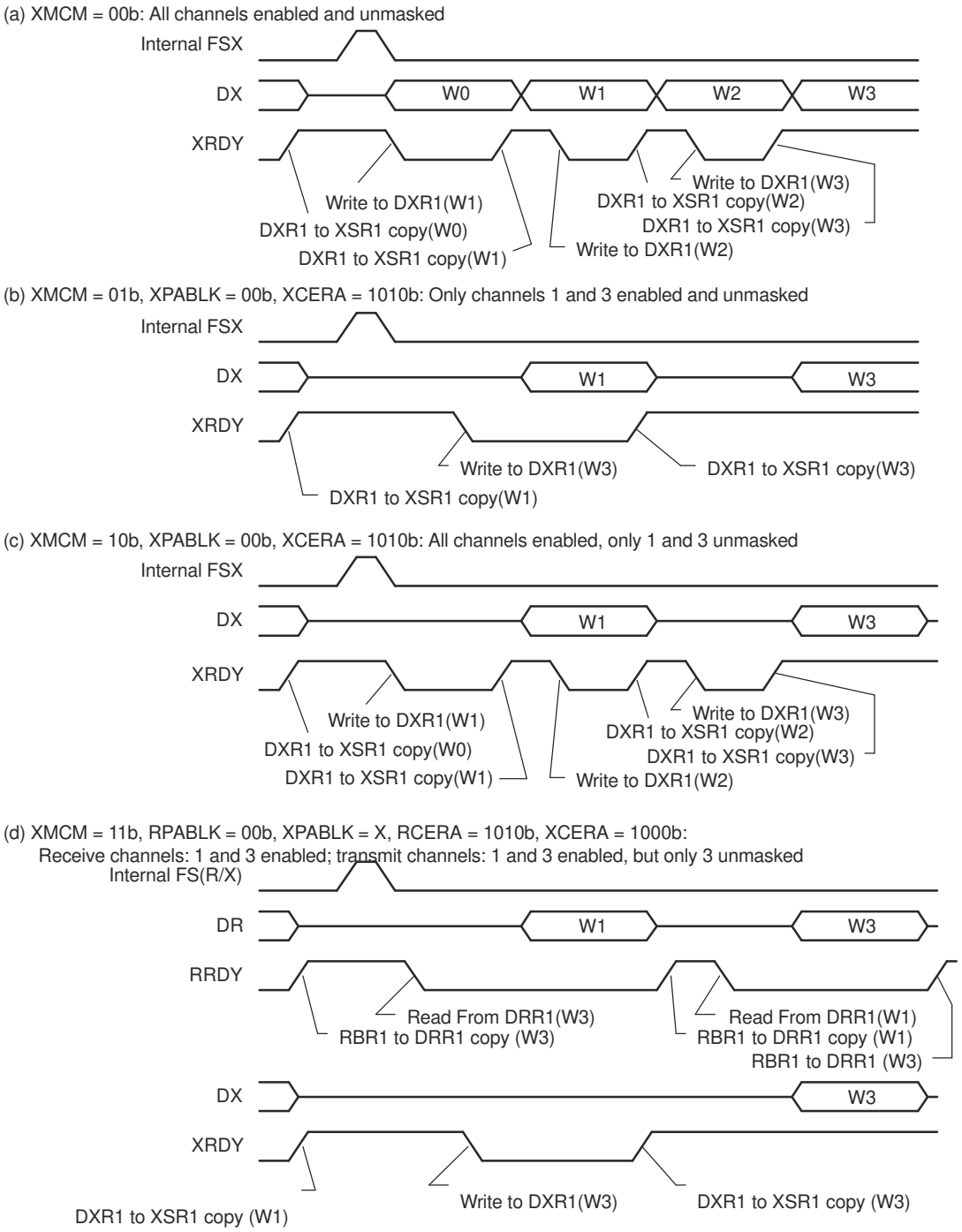



**Figure 21-35. Activity on McBSP Pins for the Possible Values of XMCM**



Copyright © 2024 Texas Instruments Incorporated

## 21.7 SPI Operation Using the Clock Stop Mode

This section explains how to use the McBSP in SPI mode.

### *21.7.1 SPI Protocol*

The SPI protocol is a master-slave configuration with one master device and one or more slave devices. The interface consists of the following four signals:

- Serial data input (also referred to as master in/slave out, or SPISOMI)
- Serial data output (also referred to as master out/slave in, or SPISIMO)
- Shift-clock (also referred to as SPICLK)
- Slave-enable signal (also referred to as $\overline{\text{SPISTE}}$)

A typical SPI interface with a single slave device is shown in Figure 21-36.

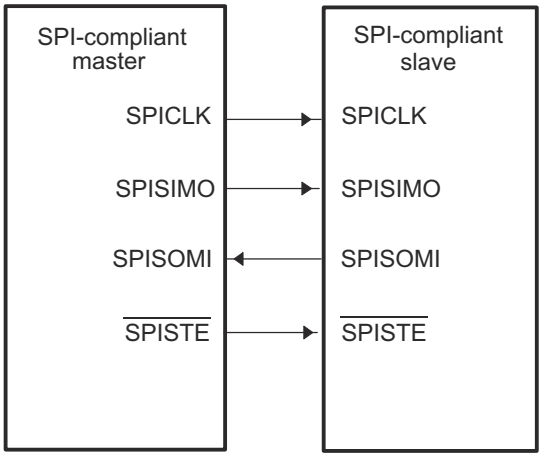



**Figure 21-36. Typical SPI Interface**

The master device controls the flow of communication by providing shift-clock and slave-enable signals. The slave-enable signal is an optional active-low signal that enables the serial data input and output of the slave device (device not sending out the clock).

In the absence of a dedicated slave-enable signal, communication between the master and slave is determined by the presence or absence of an active shift-clock. When the McBSP is operating in SPI master mode and the $\overline{\text{SPISTE}}$ signal is not used by the SPI slave port, the slave device must remain enabled at all times, and multiple slaves cannot be used.

### *21.7.2 Clock Stop Mode*

The clock stop mode of the McBSP provides compatibility with the SPI protocol. When the McBSP is configured in clock stop mode, the transmitter and receiver are internally synchronized so that the McBSP functions as an SPI master or slave device. The transmit clock signal (CLKX) corresponds to the serial clock signal (SPICLK) of the SPI protocol, while the transmit frame-synchronization signal (FSX) is used as the slave-enable signal ($\overline{\text{SPISTE}}$).

The receive clock signal (MCLKR) and receive frame-synchronization signal (FSR) are not used in the clock stop mode because these signals are internally connected to their transmit counterparts, CLKX and FSX.



Copyright © 2024 Texas Instruments Incorporated

### 21.7.3 Enable and Configure the Clock Stop Mode

The bits required to configure the McBSP as an SPI device are introduced in Table 21-14. Table 21-15 shows how the various combinations of the CLKSTP bit and the polarity bits CLKXP and CLKRP create four possible clock stop mode configurations. The timing diagrams in Section 21.7.4 show the effects of CLKSTP, CLKXP, and CLKRP.

**Table 21-14. Bits Used to Enable and Configure the Clock Stop Mode**

| Bit Field | Description |
|---|---|
| CLKSTP bits of SPCR1 | Use these bits to enable the clock stop mode and to select one of two timing variations. (See also Table 21-15.) |
| CLKXP bit of PCR | This bit determines the polarity of the CLKX signal. (See also Table 21-15.) |
| CLKRP bit of PCR | This bit determines the polarity of the MCLKR signal. (See also Table 21-15.) |
| CLKXM bit of PCR | This bit determines whether CLKX is an input signal (McBSP as slave) or an output signal (McBSP as master). |
| XPHASE bit of XCR2 | You must use a single-phase transmit frame (XPHASE = 0). |
| RPHASE bit of RCR2 | You must use a single-phase receive frame (RPHASE = 0). |
| XFRLEN1 bits of XCR1 | You must use a transmit frame length of 1 serial word (XFRLEN1 = 0). |
| RFRLEN1 bits of RCR1 | You must use a receive frame length of 1 serial word (RFRLEN1 = 0). |
| XWDLEN1 bits of XCR1 | The XWDLEN1 bits determine the transmit packet length. XWDLEN1 must be equal to RWDLEN1 because in the clock stop mode. The McBSP transmit and receive circuits are synchronized to a single clock. |
| RWDLEN1 bits of RCR1 | The RWDLEN1 bits determine the receive packet length. RWDLEN1 must be equal to XWDLEN1 because in the clock stop mode. The McBSP transmit and receive circuits are synchronized to a single clock. |

**Table 21-15. Effects of CLKSTP, CLKXP, and CLKRP on the Clock Scheme**

| Bit Settings | Clock Scheme |
|---|---|
| CLKSTP = 00b or 01b<br>CLKXP = 0 or 1<br>CLKRP = 0 or 1 | Clock stop mode disabled. Clock enabled for non-SPI mode. |
| CLKSTP = 10b<br>CLKXP = 0<br>CLKRP = 0 | Low inactive state without delay: The McBSP transmits data on the rising edge of CLKX and receives data on the falling edge of MCLKR. |
| CLKSTP = 11b<br>CLKXP = 0<br>CLKRP = 1 | Low inactive state with delay: The McBSP transmits data one-half cycle ahead of the rising edge of CLKX and receives data on the rising edge of MCLKR. |
| CLKSTP = 10b<br>CLKXP = 1<br>CLKRP = 0 | High inactive state without delay: The McBSP transmits data on the falling edge of CLKX and receives data on the rising edge of MCLKR. |
| CLKSTP = 11b<br>CLKXP = 1<br>CLKRP = 1 | High inactive state with delay: The McBSP transmits data one-half cycle ahead of the falling edge of CLKX and receives data on the falling edge of MCLKR. |



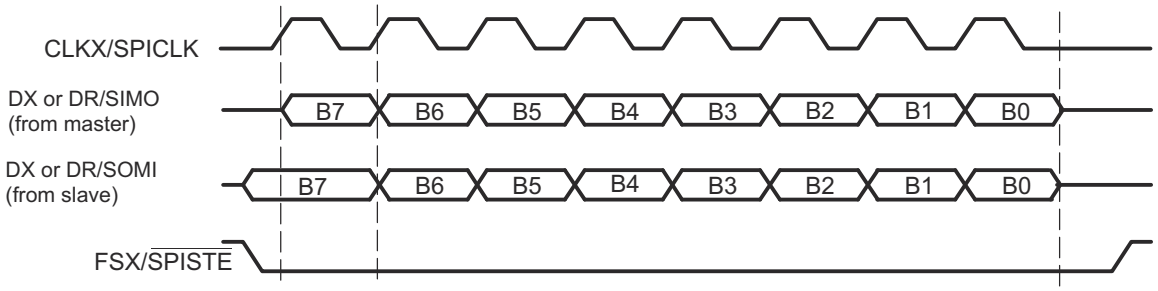
Copyright © 2024 Texas Instruments Incorporated

### 21.7.4 Clock Stop Mode Timing Diagrams

The timing diagrams for the four possible clock stop mode configurations are shown here. Notice that the frame-synchronization signal used in clock stop mode is active throughout the entire transmission as a slave-enable signal. Although the timing diagrams show 8-bit transfers, the packet length can be set to 8, 12, 16, 20, 24, or 32 bits per packet. The receive packet length is selected with the RWDLEN1 bits of RCR1, and the transmit packet length is selected with the XWDLEN1 bits of XCR1. For clock stop mode, the values of RWDLEN1 and XWDLEN1 must be the same because the McBSP transmit and receive circuits are synchronized to a single clock.

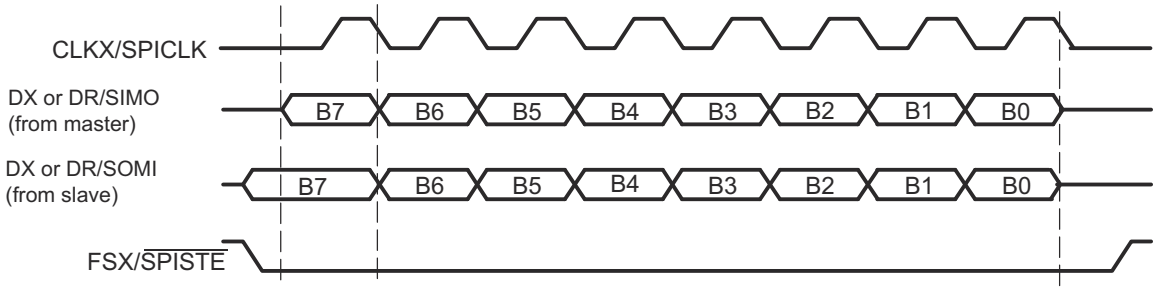
**Note**

Even if multiple words are consecutively transferred, the CLKX signal is always stopped and the FSX signal returns to the inactive state after a packet transfer. When consecutive packet transfers are performed, this leads to a minimum idle time of two bit-periods between each packet transfer.




A. If the McBSP is the SPI master (CLKXM = 1), SIMO = DX. If the McBSP is the SPI slave (CLKXM = 0), SIMO = DR.
B. If the McBSP is the SPI master (CLKXM = 1), SOMI = DR. If the McBSP is the SPI slave (CLKXM = 0), SOMI = DX.

**Figure 21-37. SPI Transfer with CLKSTP = 10b (No Clock Delay), CLKXP = 0, and CLKRP = 0**

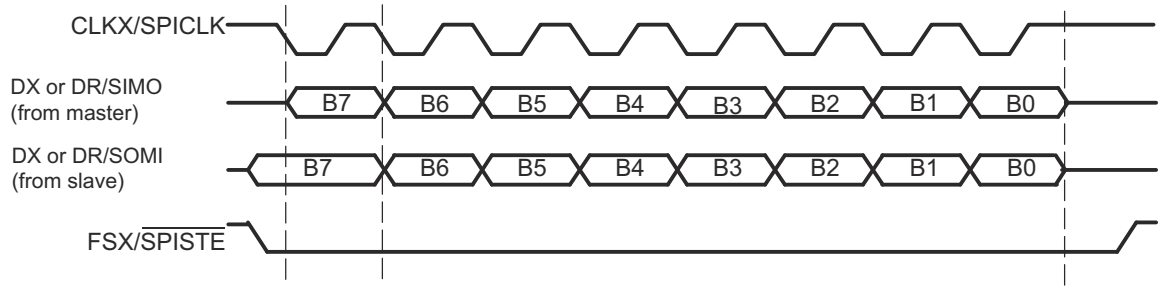



A. If the McBSP is the SPI master (CLKXM = 1), SIMO = DX. If the McBSP is the SPI slave (CLKXM = 0), SIMO = DR.
B. If the McBSP is the SPI master (CLKXM = 1), SOMI = DR. If the McBSP is the SPI slave (CLKXM = 0), SOMI = DX.

**Figure 21-38. SPI Transfer with CLKSTP = 11b (Clock Delay), CLKXP = 0, CLKRP = 1**

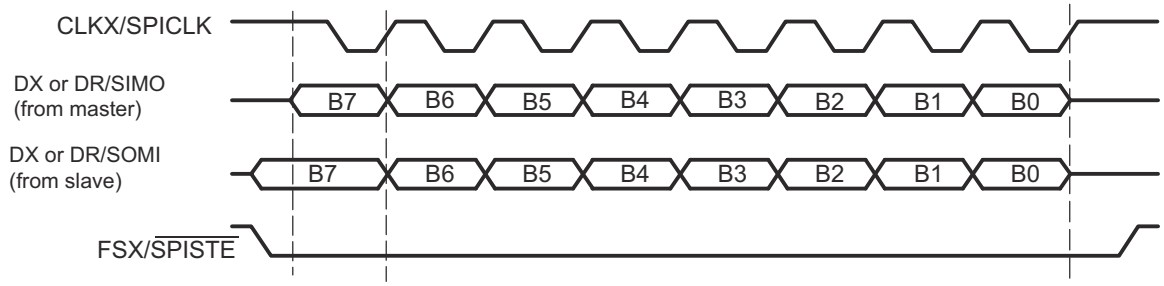


Copyright © 2024 Texas Instruments Incorporated




A. If the McBSP is the SPI master (CLKXM = 1), SIMO = DX. If the McBSP is the SPI slave (CLKXM = 0), SIMO = DR.
B. If the McBSP is the SPI master (CLKXM = 1), SOMI = DR. If the McBSP is the SPI slave (CLKXM = 0), SOMI = DX.

**Figure 21-39. SPI Transfer with CLKSTP = 10b (No Clock Delay), CLKXP = 1, and CLKRP = 0**




A. If the McBSP is the SPI master (CLKXM = 1), SIMO = DX. If the McBSP is the SPI slave (CLKXM = 0), SIMO = DR.
B. If the McBSP is the SPI master (CLKXM = 1), SOMI = DR. If the McBSP is the SPI slave (CLKXM = 0), SOMI = DX.

**Figure 21-40. SPI Transfer with CLKSTP = 11b (Clock Delay), CLKXP = 1, CLKRP = 1**



Copyright © 2024 Texas Instruments Incorporated

### 21.7.5 Procedure for Configuring a McBSP for SPI Operation

To configure the McBSP for SPI master or slave operation:

1. Place the transmitter and receiver in reset.
   Clear the transmitter reset bit (XRST = 0) in SPCR2 to reset the transmitter. Clear the receiver reset bit (RRST = 0) in SPCR1 to reset the receiver.
2. Place the sample rate generator in reset.
   Clear the sample rate generator reset bit (GRST = 0) in SPCR2 to reset the sample rate generator.
3. Program registers that affect SPI operation.
   Program the appropriate McBSP registers to configure the McBSP for proper operation as an SPI master or an SPI slave. For a list of important bits settings, see one of the following topics:
   - Section 21.7.6
   - Section 21.7.7
4. Enable the sample rate generator.
   To release the sample rate generator from reset, set the sample rate generator reset bit (GRST = 1) in SPCR2.
   Make sure that during the write to SPCR2, you only modify GRST. Otherwise, you modify the McBSP configuration you selected in the previous step.
5. Enable the transmitter and receiver.
   After the sample rate generator is released from reset, wait two sample rate generator clock periods for the McBSP logic to stabilize.
   If the CPU services the McBSP transmit and receive buffers, then you can immediately enable the transmitter (XRST = 1 in SPCR2) and enable the receiver (RRST = 1 in SPCR1).
   If the DMA controller services the McBSP transmit and receive buffers, then you must first configure the DMA controller (this includes enabling the channels that service the McBSP buffers). When the DMA controller is ready, make XRST = 1 and RRST = 1.
   In either case, make sure you only change XRST and RRST when you write to SPCR2 and SPCR1. Otherwise, you modify the bit settings you selected earlier in this procedure.
   After the transmitter and receiver are released from reset, wait two sample rate generator clock periods for the McBSP logic to stabilize.
6. If necessary, enable the frame-synchronization logic of the sample rate generator.
   After the required data acquisition setup is done (DXR[1,2] is loaded with data), set FRST = 1 if an internally generated frame-synchronization pulse is required (that is, if the McBSP is the SPI master).

### 21.7.6 McBSP as the SPI Master

An SPI interface with the McBSP used as the master is shown in Figure 21-41. When the McBSP is configured as a master, the transmit output signal (DX) is used as the SPISIMO signal of the SPI protocol and the receive input signal (DR) is used as the SPISOMI signal.

The register bit values required to configure the McBSP as a master are listed in Table 21-16. After Table 21-16 are more details about the configuration requirements.



Copyright © 2024 Texas Instruments Incorporated

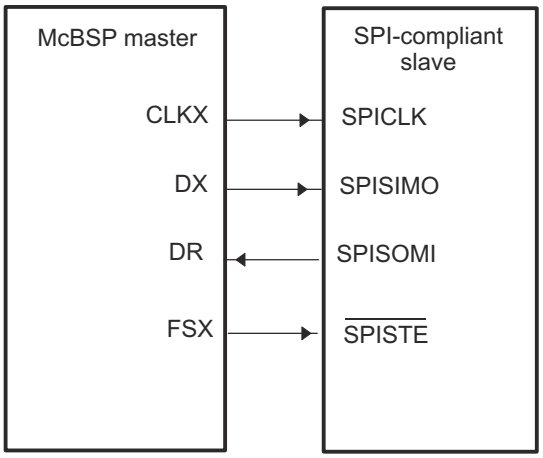



**Figure 21-41. SPI Interface with McBSP Used as Master**

**Table 21-16. Bit Values Required to Configure the McBSP as an SPI Master**

| Required Bit Setting | Description |
|---|---|
| CLKSTP = 10b or 11b | The clock stop mode (without or with a clock delay) is selected. |
| CLKXP = 0 or 1 | The polarity of CLKX as seen on the MCLKX pin is positive (CLKXP = 0) or negative (CLKXP = 1). |
| CLKRP = 0 or 1 | The polarity of MCLKR as seen on the MCLKR pin is positive (CLKRP = 0) or negative (CLKRP = 1). |
| CLKXM = 1 | The MCLKX pin is an output pin driven by the internal sample rate generator. Because CLKSTP is equal to 10b or 11b, MCLKR is driven internally by CLKX. |
| SCLKME = 0 | The clock generated by the sample rate generator (CLKG) is derived from the CPU clock. |
| CLKSM = 1 | |
| CLKGDV is a value from 1 to 255 | CLKGDV defines the divide down value for CLKG. |
| FSXM = 1 | The FSX pin is an output pin driven according to the FSGM bit. |
| FSGM = 0 | The transmitter drives a frame-synchronization pulse on the FSX pin every time data is transferred from DXR1 to XSR1. |
| FSXP = 1 | The FSX pin is active low. |
| XDATDLY = 01b | This setting provides the correct setup time on the FSX signal. |
| RDATDLY = 01b | |

When the McBSP functions as the SPI master, the McBSP controls the transmission of data by producing the serial clock signal. The clock signal on the MCLKX pin is enabled only during packet transfers. When packets are not being transferred, the MCLKX pin remains high or low depending on the polarity used.

For SPI master operation, the MCLKX pin must be configured as an output. The sample rate generator is then used to derive the CLKX signal from the CPU clock. The clock stop mode internally connects the MCLKX pin to the MCLKR signal so that no external signal connection is required on the MCLKR pin and both the transmit and receive circuits are clocked by the master clock (CLKX).

The data delay parameters of the McBSP (XDATDLY and RDATDLY) must be set to 1 for proper SPI master operation. A data delay value of 0 or 2 is undefined in the clock stop mode.

The McBSP can also provide a slave-enable signal ($\overline{\text{SPISTE}}$) on the FSX pin. If a slave-enable signal is required, the FSX pin must be configured as an output and the transmitter must be configured so that a frame-synchronization pulse is generated automatically each time a packet is transmitted (FSGM = 0). The polarity of the FSX pin is programmable high or low; however, in most cases the pin must be configured active low.

When the McBSP is configured as described for SPI master operation, the bit fields for frame-synchronization pulse width (FWID) and frame-synchronization period (FPER) are overridden, and custom frame-synchronization



Copyright © 2024 Texas Instruments Incorporated

waveforms are not allowed. To see the resulting waveform produced on the FSX pin, see the timing diagrams in Section 21.7.4. The signal becomes active before the first bit of a packet transfer, and remains active until the last bit of the packet is transferred. After the packet transfer is complete, the FSX signal returns to the inactive state.

### *21.7.7 McBSP as an SPI Slave*

An SPI with the McBSP used as a slave is shown in Figure 21-42. When the McBSP is configured as a slave, DX is used as the SPISOMI signal and DR is used as the SPISIMO signal.

The register bit values required to configure the McBSP as a slave are listed in Table 21-17. After Table 21-17 are more details about configuration requirements.

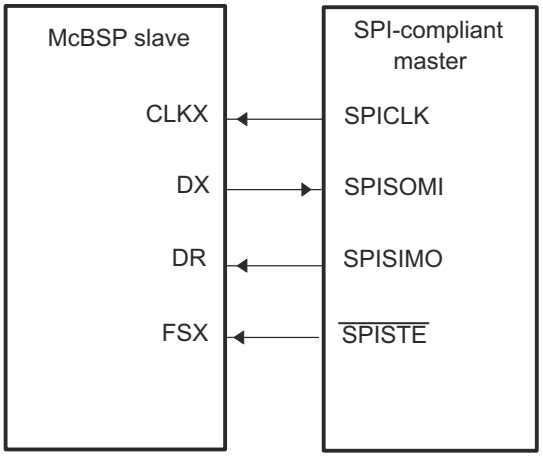



**Figure 21-42. SPI Interface With McBSP Used as Slave**

**Table 21-17. Bit Values Required to Configure the McBSP as an SPI Slave**

| Required Bit Setting | Description |
|---|---|
| CLKSTP = 10b or 11b | The clock stop mode (without or with a clock delay) is selected. |
| CLKXP = 0 or 1 | The polarity of CLKX as seen on the MCLKX pin is positive (CLKXP = 0) or negative (CLKXP = 1). |
| CLKRP = 0 or 1 | The polarity of MCLKR as seen on the MCLKR pin is positive (CLKRP = 0) or negative (CLKRP = 1). |
| CLKXM = 0 | The MCLKX pin is an input pin, so that the pin can be driven by the SPI master. Because CLKSTP = 10b or 11b, MCLKR is driven internally by CLKX. |
| SCLKME = 0<br>CLKSM = 1 | The clock generated by the sample rate generator (CLKG) is derived from the CPU clock. (The sample rate generator is used to synchronize the McBSP logic with the externally-generated master clock.) |
| CLKGDV = 1 | The sample rate generator divides the CPU clock before generating CLKG. |
| FSXM = 0 | The FSX pin is an input pin, so that the pin can be driven by the SPI master. |
| FSXP = 1 | The FSX pin is active low. |
| XDATDLY = 00b<br>RDATDLY = 00b | These bits must be 0s for SPI slave operation. |

When the McBSP is used as an SPI slave, the master clock and slave-enable signals are generated externally by a master device. Accordingly, the CLKX and FSX pins must be configured as inputs. The MCLKX pin is internally connected to the MCLKR signal, so that both the transmit and receive circuits of the McBSP are clocked by the external master clock. The FSX pin is also internally connected to the FSR signal, and no external signal connections are required on the MCLKR and FSR pins.

Although the CLKX signal is generated externally by the master and is asynchronous to the McBSP, the sample rate generator of the McBSP must be enabled for proper SPI slave operation. The sample rate generator must



Copyright © 2024 Texas Instruments Incorporated

be programmed to the maximum rate of half the CPU clock rate. The internal sample rate clock is then used to synchronize the McBSP logic to the external master clock and slave-enable signals.

The McBSP requires an active edge of the slave-enable signal on the FSX input for each transfer. This means that the master device must assert the slave-enable signal at the beginning of each transfer, and deassert the signal after the completion of each packet transfer; the slave-enable signal cannot remain active between transfers. Unlike the standard SPI, this pin cannot be tied low all the time.

The data delay parameters of the McBSP must be set to 0 for proper SPI slave operation. A value of 1 or 2 is undefined in the clock stop mode.

## 21.8 Receiver Configuration

To configure the McBSP receiver, perform the following procedure:

1. Place the McBSP/receiver in reset (see Section 21.8.2).
2. Program McBSP registers for the desired receiver operation (see Section 21.8.1).
3. Take the receiver out of reset (see Section 21.8.2).

### *21.8.1 Programming the McBSP Registers for the Desired Receiver Operation*

The following is a list of important tasks to be performed when you are configuring the McBSP receiver. Each task corresponds to one or more McBSP register bit fields.

- Global behavior:
  - Set the receiver pins to operate as McBSP pins.
  - Enable/disable the digital loopback mode.
  - Enable/disable the clock stop mode.
  - Enable/disable the receive multichannel selection mode.
- Data behavior:
  - Choose 1 or 2 phases for the receive frame.
  - Set the receive word length(s).
  - Set the receive frame length.
  - Enable/disable the receive frame-synchronization ignore function.
  - Set the receive companding mode.
  - Set the receive data delay.
  - Set the receive sign-extension and justification mode.
  - Set the receive interrupt mode.
- Frame-synchronization behavior:
  - Set the receive frame-synchronization mode.
  - Set the receive frame-synchronization polarity.
  - Set the sample rate generator (SRG) frame-synchronization period and pulse width.
- Clock behavior:
  - Set the receive clock mode.
  - Set the receive clock polarity.
  - Set the SRG clock divide-down value.
  - Set the SRG clock synchronization mode.
  - Set the SRG clock mode (choose an input clock).
  - Set the SRG input clock polarity.



Copyright © 2024 Texas Instruments Incorporated

### 21.8.2 Resetting and Enabling the Receiver

The first step of the receiver configuration procedure is to reset the receiver, and the last step is to enable the receiver (to take the receiver out of reset). Table 21-18 describes the bits used for both of these steps.

**Table 21-18. Register Bits Used to Reset or Enable the McBSP Receiver Field Descriptions**

| Register | Bit | Field | Value | Description |
|---|---|---|---|---|
| SPCR2 | 7 | FRST | | Frame-synchronization logic reset |
| | | | 0 | Frame-synchronization logic is reset. The sample rate generator does not generate frame-synchronization signal FSG, even if GRST = 1. |
| | | | 1 | If GRST = 1, frame-synchronization signal FSG is generated after (FPER + 1) number of CLKG clock cycles; all frame counters are loaded with their programmed values. |
| SPCR2 | 6 | GRST | | Sample rate generator reset |
| | | | 0 | Sample rate generator is reset. If GRST = 0 due to a DSP reset, CLKG is driven by the CPU clock divided by 2, and FSG is driven low (inactive). If GRST = 0 due to program code, CLKG and FSG are both driven low (inactive). |
| | | | 1 | Sample rate generator is enabled. CLKG is driven according to the configuration programmed in the sample rate generator registers (SRGR[1,2]). If FRST = 1, the generator also generates the frame-synchronization signal FSG as programmed in the sample rate generator registers. |
| SPCR1 | 0 | RRST | | Receiver reset |
| | | | 0 | The serial port receiver is disabled and in the reset state. |
| | | | 1 | The serial port receiver is enabled. |

#### 21.8.2.1 Reset Considerations

The serial port can be reset in the following two ways:

1. The DSP reset ($\overline{\text{XRS}}$ signal driven low) places the receiver, transmitter, and sample rate generator in reset. When the device reset is removed ($\overline{\text{XRS}}$ signal released), GRST = FRST = RRST = XRST = 0 keep the entire serial port in the reset state, provided the McBSP clock is turned on.
2. The serial port transmitter and receiver can be reset directly using the RRST and XRST bits in the serial port control registers. The sample rate generator can be reset directly using the GRST bit in SPCR2.

Table 21-19 shows the state of McBSP pins when the serial port is reset due to a device reset and a direct receiver/transmitter reset.

For more details about McBSP reset conditions and effects, see Section 21.10.2.

**Table 21-19. Reset State of Each McBSP Pin**

| Pin | Possible States[1] | State Forced By Device Reset | State Forced By Receiver Reset (RRST = 0 and GRST = 1) |
|---|---|---|---|
| MDRx | I | GPIO Input | Input |
| MCLKRx | I/O/Z | GPIO Input | Known state, if input; MCLKR running, if output |
| MFSRx | I/O/Z | GPIO Input | Known state, if input; FSRP inactive state, if output |
| | | | Transmitter reset (XRST = 0 and GRST = 1) |
| MDXx | O/Z | GPIO Input | Low impedance after transmit bit clock provided |
| MCLKXx | I/O/Z | GPIO Input | Known state, if input; CLKX running, if output |
| MFSXx | I/O/Z | GPIO Input | Known state, if input; FSXP inactive state, if output |

(1) In Possible States column, I = input, O = output, Z = high impedance



Copyright © 2024 Texas Instruments Incorporated

### 21.8.3 Set the Receiver Pins to Operate as McBSP Pins

To configure a pin for McBSP functioning, configure the bits of the GPxMUXn register appropriately. In addition to this, bits 12 and 13 of the PCR register must be set to 0. These bits are defined as reserved.

### 21.8.4 Digital Loopback Mode

The DLB bit determines whether the digital loopback mode is on. DLB is described in Table 21-20.

**Table 21-20. Register Bit Used to Enable/Disable the Digital Loopback Mode**

| Register | Bit | Name | Function | | Type | Reset Value |
|---|---|---|---|---|---|---|
| SPCR1 | 15 | DLB | Digital loopback mode | | R/W | 0 |
| | | | DLB = 0 | Digital loopback mode is disabled. | | |
| | | | DLB = 1 | Digital loopback mode is enabled. | | |

In the digital loopback mode, the receive signals are connected internally through multiplexers to the corresponding transmit signals, as shown in Table 21-21. This mode allows testing of serial port code with a single DSP device; the McBSP receives the data the McBSP transmits.

**Table 21-21. Receive Signals Connected to Transmit Signals in Digital Loopback Mode**

| This Receive Signal | Is Fed Internally by This Transmit Signal |
|---|---|
| MDR (receive data) | MDX (transmit data) |
| MFSR (receive frame synchronization) | MFSX (transmit frame synchronization) |
| MCLKR (receive clock) | MCLKX (transmit clock) |

### 21.8.5 Clock Stop Mode

The CLKSTP bits determine whether the clock stop mode is on. CLKSTP is described in Table 21-22.

**Table 21-22. Register Bits Used to Enable/Disable the Clock Stop Mode**

| Register | Bit | Name | Function | | Type | Reset Value |
|---|---|---|---|---|---|---|
| SPCR1 | 12-11 | CLKSTP | Clock stop mode | | R/W | 00 |
| | | | CLKSTP = 0Xb | Clock stop mode disabled; normal clocking for non-SPI mode | | |
| | | | CLKSTP = 10b | Clock stop mode enabled, without clock delay | | |
| | | | CLKSTP = 11b | Clock stop mode enabled, with clock delay | | |

The clock stop mode supports the SPI master-slave protocol. If you do not plan to use the SPI protocol, you can clear CLKSTP to disable the clock stop mode.

In the clock stop mode, the clock stops at the end of each data transfer. At the beginning of each data transfer, the clock starts immediately (CLKSTP = 10b) or after a half-cycle delay (CLKSTP = 11b). The CLKXP bit determines whether the starting edge of the clock on the MCLKX pin is rising or falling. The CLKRP bit determines whether receive data is sampled on the rising or falling edge of the clock shown on the MCLKR pin.

Table 21-23 summarizes the impact of CLKSTP, CLKXP, and CLKRP on serial port operation. In the clock stop mode, the receive clock is tied internally to the transmit clock, and the receive frame-synchronization signal is tied internally to the transmit frame-synchronization signal.



Copyright © 2024 Texas Instruments Incorporated

**Table 21-23. Effects of CLKSTP, CLKXP, and CLKRP on the Clock Scheme**

| Bit Settings | Clock Scheme |
|---|---|
| CLKSTP = 00b or 01b<br>CLKXP = 0 or 1<br>CLKRP = 0 or 1 | Clock stop mode disabled. Clock enabled for non-SPI mode. |
| CLKSTP = 10b<br>CLKXP = 0<br>CLKRP = 0 | Low inactive state without delay: The McBSP transmits data on the rising edge of CLKX and receives data on the falling edge of MCLKR. |
| CLKSTP = 11b<br>CLKXP = 0<br>CLKRP = 1 | Low inactive state with delay: The McBSP transmits data one-half cycle ahead of the rising edge of CLKX and receives data on the rising edge of MCLKR. |
| CLKSTP = 10b<br>CLKXP = 1<br>CLKRP = 0 | High inactive state without delay: The McBSP transmits data on the falling edge of CLKX and receives data on the rising edge of MCLKR. |
| CLKSTP = 11b<br>CLKXP = 1<br>CLKRP = 1 | High inactive state with delay: The McBSP transmits data one-half cycle ahead of the falling edge of CLKX and receives data on the falling edge of MCLKR. |

### *21.8.6 Receive Multichannel Selection Mode*

The RMCM bit determines whether the receive multichannel selection mode is on. RMCM is described in Table 21-24. For more details, see Section 21.6.6.

**Table 21-24. Register Bit Used to Enable/Disable the Receive Multichannel Selection Mode**

| Register | Bit | Name | Function | | Type | Reset Value |
|---|---|---|---|---|---|---|
| MCR1 | 0 | RMCM | Receive multichannel selection mode | | R/W | 0 |
| | | | RMCM = 0 | The mode is disabled.<br>All 128 channels are enabled. | | |
| | | | RMCM = 1 | The mode is enabled.<br>Channels can be individually enabled or disabled.<br>The only channels enabled are those selected in the appropriate receive channel enable registers (RCERs). The way channels are assigned to the RCERs depends on the number of receive channel partitions (2 or 8), as defined by the RMCME bit. | | |

### *21.8.7 Receive Frame Phases*

The RPHASE bit (see Table 21-25) determines whether the receive data frame has one or two phases.

**Table 21-25. Register Bit Used to Choose One or Two Phases for the Receive Frame**

| Register | Bit | Name | Function | | Type | Reset Value |
|---|---|---|---|---|---|---|
| RCR2 | 15 | RPHASE | Receive phase number | | R/W | 0 |
| | | | Specifies whether the receive frame has 1 or 2 phases. | | | |
| | | | RPHASE = 0 | Single-phase frame | | |
| | | | RPHASE = 1 | Dual-phase frame | | |



Copyright © 2024 Texas Instruments Incorporated

### 21.8.8 Receive Word Lengths

The RWDLEN1 and RWDLEN2 bit fields (see Table 21-26) determine how many bits are in each serial word in phase 1 and in phase 2, respectively, of the receive data frame.

**Table 21-26. Register Bits Used to Set the Receive Word Lengths**

| Register | Bit | Name | Function | | Type | Reset Value |
|---|---|---|---|---|---|---|
| RCR1 | 7-5 | RWDLEN1 | Receive word length 1 | | R/W | 000 |
| | | | Specifies the length of every serial word in phase 1 of the receive frame. | | | |
| | | | RWDLEN1 = 000 | 8 bits | | |
| | | | RWDLEN1 = 001 | 12 bits | | |
| | | | RWDLEN1 = 010 | 16 bits | | |
| | | | RWDLEN1 = 011 | 20 bits | | |
| | | | RWDLEN1 = 100 | 24 bits | | |
| | | | RWDLEN1 = 101 | 32 bits | | |
| | | | RWDLEN1 = 11X | Reserved | | |
| RCR2 | 7-5 | RWDLEN2 | Receive word length 2 | | R/W | 000 |
| | | | If a dual-phase frame is selected, RWDLEN2 specifies the length of every serial word in phase 2 of the frame. | | | |
| | | | RWDLEN2 = 000 | 8 bits | | |
| | | | RWDLEN2 = 001 | 12 bits | | |
| | | | RWDLEN2 = 010 | 16 bits | | |
| | | | RWDLEN2 = 011 | 20 bits | | |
| | | | RWDLEN2 = 100 | 24 bits | | |
| | | | RWDLEN2 = 101 | 32 bits | | |
| | | | RWDLEN2 = 11X | Reserved | | |

#### 21.8.8.1 Word Length Bits

Each frame can have one or two phases, depending on the value that you load into the RPHASE bit. If a single-phase frame is selected, RWDLEN1 selects the length for every serial word received in the frame. If a dual-phase frame is selected, RWDLEN1 determines the length of the serial words in phase 1 of the frame and RWDLEN2 determines the word length in phase 2 of the frame.



Copyright © 2024 Texas Instruments Incorporated

### 21.8.9 Receive Frame Length

The RFRLEN1 and RFRLEN2 bit fields (see Table 21-27) determine how many serial words are in phase 1 and in phase 2, respectively, of the receive data frame.

**Table 21-27. Register Bits Used to Set the Receive Frame Length**

| Register | Bit | Name | Function | | Type | Reset Value |
|---|---|---|---|---|---|---|
| RCR1 | 14-8 | RFRLEN1 | Receive frame length 1 | | R/W | 000 0000 |
| | | | (RFRLEN1 + 1) is the number of serial words in phase 1 of the receive frame. | | | |
| | | | RFRLEN1 = 000 0000 | 1 word in phase 1 | | |
| | | | RFRLEN1 = 000 0001 | 2 words in phase 1 | | |
| | | | \| | \| | | |
| | | | \| | \| | | |
| | | | RFRLEN1 = 111 1111 | 128 words in phase 1 | | |
| RCR2 | 14-8 | RFRLEN2 | Receive frame length 2 | | R/W | 000 0000 |
| | | | If a dual-phase frame is selected, (RFRLEN2 + 1) is the number of serial words in phase 2 of the receive frame. | | | |
| | | | RFRLEN2 = 000 0000 | 1 word in phase 2 | | |
| | | | RFRLEN2 = 000 0001 | 2 words in phase 2 | | |
| | | | \| | \| | | |
| | | | \| | \| | | |
| | | | RFRLEN2 = 111 1111 | 128 words in phase 2 | | |

#### 21.8.9.1 Selected Frame Length

The receive frame length is the number of serial words in the receive frame. Each frame can have one or two phases, depending on value that you load into the RPHASE bit.

If a single-phase frame is selected (RPHASE = 0), the frame length is equal to the length of phase 1. If a dual-phase frame is selected (RPHASE = 1), the frame length is the length of phase 1 plus the length of phase 2.

The 7-bit RFRLEN fields allow up to 128 words per phase. See Table 21-28 for a summary of how to calculate the frame length. This length corresponds to the number of words or logical time slots or channels per frame-synchronization pulse.

Program the RFRLEN fields with [*w minus 1*], where *w* represents the number of words per phase. For the example, if you want a phase length of 128 words in phase 1, load 127 into RFRLEN1.

**Table 21-28. How to Calculate the Length of the Receive Frame**

| RPHASE | RFRLEN1 | RFRLEN2 | Frame Length |
|---|---|---|---|
| 0 | 0 ≤ RFRLEN1 ≤ 127 | Don't care | (RFRLEN1 + 1) words |
| 1 | 0 ≤ RFRLEN1 ≤ 127 | 0 ≤ RFRLEN2 ≤ 127 | (RFRLEN1 + 1) + (RFRLEN2 + 1) words |



Copyright © 2024 Texas Instruments Incorporated

### 21.8.10 Receive Frame-Synchronization Ignore Function

The RFIG bit (see Table 21-29) controls the receive frame-synchronization ignore function.

**Table 21-29. Register Bit Used to Enable/Disable the Receive Frame-Synchronization Ignore Function**

| Register | Bit | Name | Function | | Type | Reset Value |
|---|---|---|---|---|---|---|
| RCR2 | 2 | RFIG | Receive frame-synchronization ignore | | R/W | 0 |
| | | | RFIG = 0 | An unexpected receive frame-synchronization pulse causes the McBSP to restart the frame transfer. | | |
| | | | RFIG = 1 | The McBSP ignores unexpected receive frame-synchronization pulses. | | |

#### 21.8.10.1 Unexpected Frame-Synchronization Pulses and the Frame-Synchronization Ignore Function

If a frame-synchronization pulse starts the transfer of a new frame before the current frame is fully received, this pulse is treated as an unexpected frame-synchronization pulse.

When RFIG = 1, reception continues, ignoring the unexpected frame-synchronization pulses.

When RFIG = 0, an unexpected FSR pulse causes the McBSP to discard the contents of RSR[1,2] in favor of the new incoming data. Therefore, if RFIG = 0 and an unexpected frame-synchronization pulse occurs, the serial port:

1. Aborts the current data transfer
2. Sets RSYNCERR in SPCR1 to 1
3. Begins the transfer of a new data word

For more details about the frame-synchronization error condition, see Section 21.5.3.



Copyright © 2024 Texas Instruments Incorporated

#### 21.8.10.2 Examples of Effects of RFIG

Figure 21-43 shows an example in which word B is interrupted by an unexpected frame-synchronization pulse when (R/X)FIG = 0. In the case of reception, the reception of B is aborted (B is lost), and a new data word C in this example) is received after the appropriate data delay. This condition is a receive synchronization error, which sets the RSYNCERR bit.

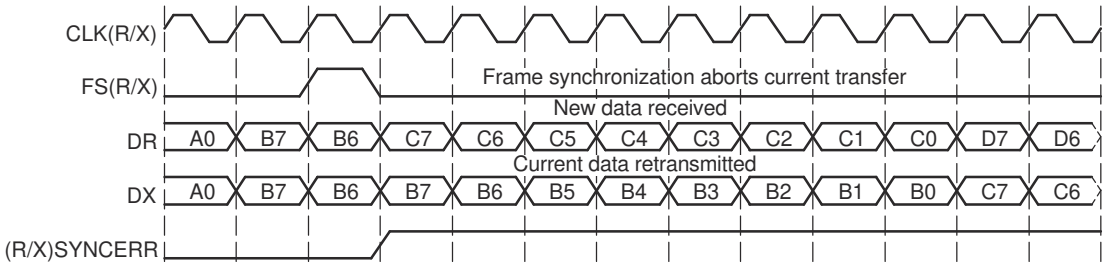



**Figure 21-43. Unexpected Frame-Synchronization Pulse With (R/X)FIG = 0**

In contrast with Figure 21-43, Figure 21-44 shows McBSP operation when unexpected frame-synchronization signals are ignored (when (R/X)FIG = 1). Here, the transfer of word B is not affected by an unexpected pulse.

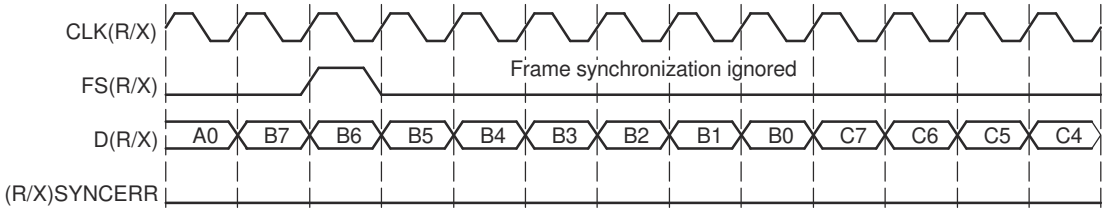



**Figure 21-44. Unexpected Frame-Synchronization Pulse With (R/X)FIG = 1**

### *21.8.11 Receive Companding Mode*

The RCOMPAND bits (see Table 21-30) determine whether companding or another data transfer option is chosen for McBSP reception.

**Table 21-30. Register Bits Used to Set the Receive Companding Mode**

| Register | Bit | Name | Function | | Type | Reset Value |
|---|---|---|---|---|---|---|
| RCR2 | 4-3 | RCOMPAND | Receive companding mode | | R/W | 00 |
| | | | Modes other than 00b are enabled only when the appropriate RWDLEN is 000b, indicating 8-bit data. | | | |
| | | | RCOMPAND = 00 | No companding, any size data, MSB received first | | |
| | | | RCOMPAND = 01 | No companding, 8-bit data, LSB received first (for details, see Section 21.8.11.4). | | |
| | | | RCOMPAND = 10 | µ-law companding, 8-bit data, MSB received first | | |
| | | | RCOMPAND = 11 | A-law companding, 8-bit data, MSB received first | | |



Copyright © 2024 Texas Instruments Incorporated

#### 21.8.11.1 Companding

Companding (COMpressing and exPANDing) hardware allows compression and expansion of data in either μ-law or A-law format. The companding standard employed in the United States and Japan is μ-law. The European companding standard is referred to as A-law. The specifications for μ-law and A-law log PCM are part of the CCITT G.711 recommendation.

A-law and μ-law allow 13 bits and 14 bits of dynamic range, respectively. Any values outside this range are set to the most positive or most negative value. Thus, for companding to work best, the data transferred to and from the McBSP via the CPU or DMA controller must be at least 16 bits wide.

The μ-law and A-law formats both encode data into 8-bit code words. Companded data is always 8 bits wide; the appropriate word length bits (RWDLEN1, RWDLEN2, XWDLEN1, XWDLEN2) must therefore be set to 0, indicating an 8-bit wide serial data stream. If companding is enabled and either of the frame phases does not have an 8-bit word length, companding continues as if the word length is 8 bits.

Figure 21-45 illustrates the companding processes. When companding is chosen for the transmitter, compression occurs during the process of copying data from DXR1 to XSR1. The transmit data is encoded according to the specified companding law (A-law or μ-law). When companding is chosen for the receiver, expansion occurs during the process of copying data from RBR1 to DRR1. The receive data is decoded to 2's-complement format.

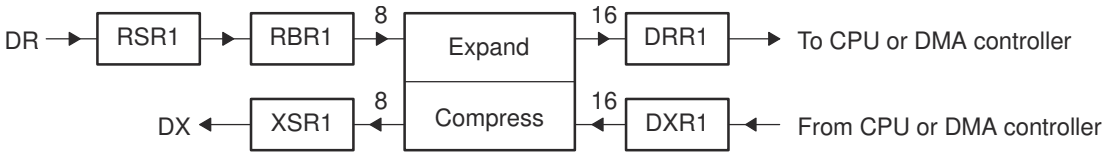



**Figure 21-45. Companding Processes for Reception and for Transmission**

#### 21.8.11.2 Format of Expanded Data

For reception, the 8-bit compressed data in RBR1 is expanded to left-justified 16-bit data in DRR1. The RJUST bit of SPCR1 is ignored when companding is used.

#### 21.8.11.3 Companding Internal Data

If the McBSP is otherwise unused (the serial port transmit and receive sections are reset), the companding hardware can compand internal data. See Section 21.3.2.2.

#### 21.8.11.4 Option to Receive LSB First

Normally, the McBSP transmits or receives all data with the most significant bit (MSB) first. However, certain 8-bit data protocols (that do not use companded data) require the least significant bit (LSB) to be transferred first. If you set RCOMPAND = 01b in RCR2, the bit ordering of 8-bit words is reversed during reception. Similar to companding, this feature is enabled only if the appropriate word length bits are set to 0, indicating that 8-bit words are to be transferred serially. If either phase of the frame does not have an 8-bit word length, the McBSP assumes the word length is eight bits and LSB-first ordering is done.



Copyright © 2024 Texas Instruments Incorporated

### 21.8.12 Receive Data Delay

The RDATDLY bits (see Table 21-31) determine the length of the data delay for the receive frame.

**Table 21-31. Register Bits Used to Set the Receive Data Delay**

| Register | Bit | Name | Function | | Type | Reset Value |
|---|---|---|---|---|---|---|
| RCR2 | 1-0 | RDATDLY | Receive data delay | | R/W | 00 |
| | | | RDATDLY = 00 | 0-bit data delay | | |
| | | | RDATDLY = 01 | 1-bit data delay | | |
| | | | RDATDLY = 10 | 2-bit data delay | | |
| | | | RDATDLY = 11 | Reserved | | |

#### 21.8.12.1 Data Delay

The start of a frame is defined by the first clock cycle in which frame synchronization is found to be active. The beginning of actual data reception or transmission with respect to the start of the frame can be delayed if required. This delay is called data delay.

RDATDLY specifies the data delay for reception. The range of programmable data delay is zero to two bit-clocks (RDATDLY = 00b-10b), as described in Table 21-31 and shown in Figure 21-46. In this figure, the data transferred is an 8-bit value with bits labeled B7, B6, B5, and so on. Typically a 1-bit delay is selected, because data often follows a 1-cycle active frame-synchronization pulse.

#### 21.8.12.2 0-Bit Data Delay

Normally, a frame-synchronization pulse is detected or sampled with respect to an edge of internal serial clock CLK(R/X). Thus, on the following cycle or later (depending on the data delay value), data may be received or transmitted. However, in the case of 0-bit data delay, the data must be ready for reception and/or transmission on the same serial clock cycle.

For reception, this problem is solved because receive data is sampled on the first falling edge of MCLKR where an active-high internal FSR is detected. However, data transmission must begin on the rising edge of the internal CLKX clock that generated the frame synchronization. Therefore, the first data bit is assumed to be present in XSR1, and thus on DX. The transmitter then asynchronously detects the frame-synchronization signal (FSX) going active high and immediately starts driving the first bit to be transmitted on the DX pin.

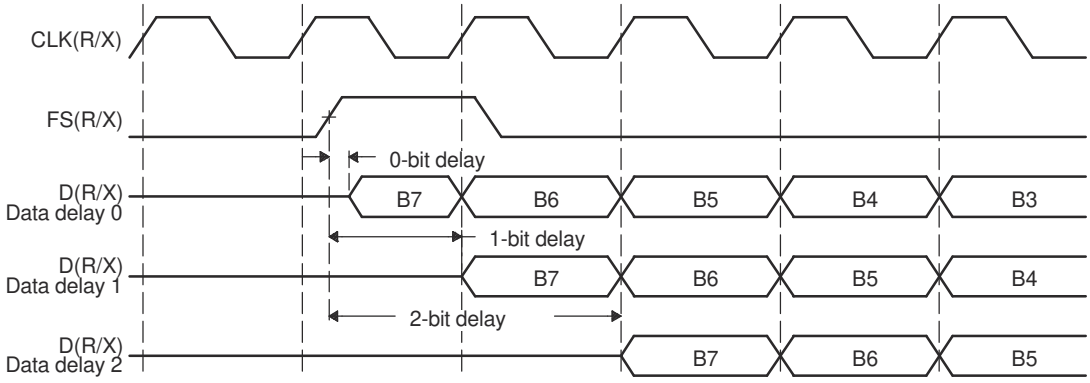



**Figure 21-46. Range of Programmable Data Delay**



Copyright © 2024 Texas Instruments Incorporated

#### 21.8.12.3 2-Bit Data Delay

A data delay of two bit periods allows the serial port to interface to different types of T1 framing devices where the data stream is preceded by a framing bit. During reception of such a stream with data delay of two bits (framing bit appears after a 1-bit delay and data appears after a 2-bit delay), the serial port essentially discards the framing bit from the data stream, as shown in Figure 21-47. In this figure, the data transferred is an 8-bit value with bits labeled B7, B6, B5, and so on.

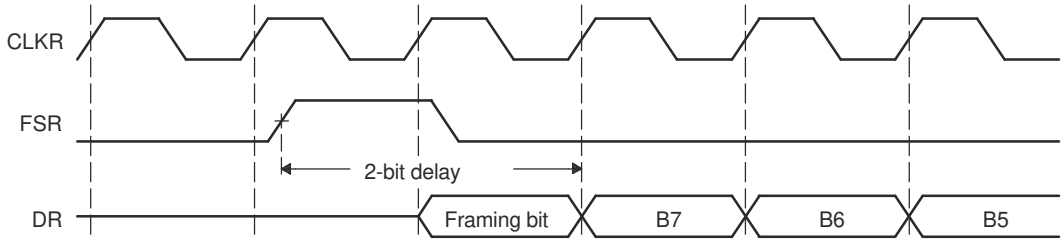



**Figure 21-47. 2-Bit Data Delay Used to Skip a Framing Bit**



Copyright © 2024 Texas Instruments Incorporated

### 21.8.13 Receive Sign-Extension and Justification Mode

The RJUST bits (see Table 21-32) determine whether data received by the McBSP is sign-extended and how the data is justified.

**Table 21-32. Register Bits Used to Set the Receive Sign-Extension and Justification Mode**

| Register | Bit | Name | Function | | Type | Reset Value |
|---|---|---|---|---|---|---|
| SPCR1 | 14-13 | RJUST | Receive sign-extension and justification mode | | R/W | 00 |
| | | | RJUST = 00 | Right justify data and zero fill MSBs in DRR[1,2] | | |
| | | | RJUST = 01 | Right justify data and sign extend the data into the MSBs in DRR[1,2] | | |
| | | | RJUST = 10 | Left justify data and zero fill LSBs in DRR[1,2] | | |
| | | | RJUST = 11 | Reserved | | |

#### 21.8.13.1 Sign-Extension and the Justification

RJUST in SPCR1 selects whether data in RBR[1,2] is right- or left-justified (with respect to the MSB) in DRR[1,2] and whether unused bits in DRR[1,2] are filled with zeros or with sign bits.

Table 21-33 and Table 21-34 show the effects of various RJUST values. The first table shows the effect on an example 12-bit receive-data value ABCh. The second table shows the effect on an example 20-bit receive-data value ABCDEh.

**Table 21-33. Example: Use of RJUST Field With 12-Bit Data Value ABCh**

| RJUST | Justification | Extension | Value in DRR2 | Value in DRR1 |
|---|---|---|---|---|
| 00b | Right | Zero fill MSBs | 0000h | 0ABCh |
| 01b | Right | Sign extend data into MSBs | FFFFh | FABCh |
| 10b | Left | Zero fill LSBs | 0000h | ABC0h |
| 11b | Reserved | Reserved | Reserved | Reserved |

**Table 21-34. Example: Use of RJUST Field With 20-Bit Data Value ABCDEh**

| RJUST | Justification | Extension | Value in DRR2 | Value in DRR1 |
|---|---|---|---|---|
| 00b | Right | Zero fill MSBs | 000Ah | BCDEh |
| 01b | Right | Sign extend data into MSBs | FFFAh | BCDEh |
| 10b | Left | Zero fill LSBs | ABCDh | E000h |
| 11b | Reserved | Reserved | Reserved | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 21.8.14 Receive Interrupt Mode

The RINTM bits (see Table 21-35) determine which event generates a receive interrupt request to the CPU.

The receive interrupt (RINT) informs the CPU of changes to the serial port status. Four options exist for configuring this interrupt. The options are set by the receive interrupt mode bits, RINTM, in SPCR1.

**Table 21-35. Register Bits Used to Set the Receive Interrupt Mode**

| Register | Bit | Name | Function | | Type | Reset Value |
|---|---|---|---|---|---|---|
| SPCR1 | 5-4 | RINTM | Receive interrupt mode | | R/W | 00 |
| | | | RINTM = 00 | RINT generated when RRDY changes from 0 to 1. Interrupt on every serial word by tracking the RRDY bit in SPCR1. Regardless of the value of RINTM, RRDY can be read to detect the RRDY = 1 condition. | | |
| | | | RINTM = 01 | RINT generated by an end-of-block or end-of-frame condition in the receive multichannel selection mode. In the multichannel selection mode, interrupt after every 16-channel block boundary has been crossed within a frame and at the end of the frame. For details, see Section 21.6.8. In any other serial transfer case, this setting is not applicable and, therefore, no interrupts are generated. | | |
| | | | RINTM = 10 | RINT generated by a new receive frame-synchronization pulse. Interrupt on detection of receive frame-synchronization pulses. This generates an interrupt even when the receiver is in the reset state. This is done by synchronizing the incoming frame-synchronization pulse to the CPU clock and sending the pulse to the CPU using RINT. | | |
| | | | RINTM = 11 | RINT generated when RSYNCERR is set. Interrupt on frame-synchronization error. Regardless of the value of RINTM, RSYNCERR can be read to detect this condition. For information on using RSYNCERR, see Section 21.5.3. | | |

### 21.8.15 Receive Frame-Synchronization Mode

The bits described in Table 21-36 determine the source for receive frame synchronization and the function of the FSR pin.

#### 21.8.15.1 Receive Frame-Synchronization Modes

Table 21-37 shows how you can select various sources to provide the receive frame-synchronization signal and the effect on the FSR pin. The polarity of the signal on the FSR pin is determined by the FSRP bit.

In digital loopback mode (DLB = 1), the transmit frame-synchronization signal is used as the receive frame-synchronization signal.

Also in the clock stop mode, the internal receive clock signal (MCLKR) and the internal receive frame-synchronization signal (FSR) are internally connected to their transmit counterparts, CLKX and FSX.



Copyright © 2024 Texas Instruments Incorporated

**Table 21-36. Register Bits Used to Set the Receive Frame Synchronization Mode**

| Register | Bit | Name | Function | | Type | Reset Value |
|---|---|---|---|---|---|---|
| PCR | 10 | FSRM | Receive frame-synchronization mode | | R/W | 0 |
| | | | FSRM = 0 | Receive frame synchronization is supplied by an external source via the FSR pin. | | |
| | | | FSRM = 1 | Receive frame synchronization is supplied by the sample rate generator. FSR is an output pin reflecting internal FSR, except when GSYNC = 1 in SRGR2. | | |
| SRGR2 | 15 | GSYNC | Sample rate generator clock synchronization mode | | R/W | 0 |
| | | | If the sample rate generator creates a frame-synchronization signal (FSG) that is derived from an external input clock, the GSYNC bit determines whether FSG is kept synchronized with pulses on the FSR pin. | | | |
| | | | GSYNC = 0 | No clock synchronization is used: CLKG oscillates without adjustment, and FSG pulses every (FPER + 1) CLKG cycles. | | |
| | | | GSYNC = 1 | Clock synchronization is used. When a pulse is detected on the FSR pin:<br>• CLKG is adjusted as necessary so that CLKG is synchronized with the input clock on the MCLKR pin.<br>• FSG pulses FSG only pulses in response to a pulse on the FSR pin. The frame-synchronization period defined in FPER is ignored.<br>For more details, see Section 21.4.3. | | |
| SPCR1 | 15 | DLB | Digital loopback mode | | R/W | 0 |
| | | | DLB = 0 | Digital loopback mode is disabled. | | |
| | | | DLB = 1 | Digital loopback mode is enabled. The receive signals, including the receive frame-synchronization signal, are connected internally through multiplexers to the corresponding transmit signals. | | |
| SPCR1 | 12-11 | CLKSTP | Clock stop mode | | R/W | 00 |
| | | | CLKSTP = 0Xb | Clock stop mode disabled; normal clocking for non-SPI mode. | | |
| | | | CLKSTP = 10b | Clock stop mode enabled without clock delay. The internal receive clock signal (MCLKR) and the internal receive frame-synchronization signal (FSR) are internally connected to their transmit counterparts, CLKX and FSX. | | |
| | | | CLKSTP = 11b | Clock stop mode enabled with clock delay. The internal receive clock signal (MCLKR) and the internal receive frame-synchronization signal (FSR) are internally connected to their transmit counterparts, CLKX and FSX. | | |



Copyright © 2024 Texas Instruments Incorporated

**Table 21-37. Select Sources to Provide the Receive Frame-Synchronization Signal and the Effect on the FSR Pin**

| DLB | FSRM | GSYNC | Source of Receive Frame Synchronization | FSR Pin Status |
|---|---|---|---|---|
| 0 | 0 | 0 or 1 | An external frame-synchronization signal enters the McBSP through the FSR pin. The signal is then inverted as determined by FSRP before being used as internal FSR. | Input |
| 0 | 1 | 0 | Internal FSR is driven by the sample rate generator frame-synchronization signal (FSG). | Output. FSG is inverted as determined by FSRP before being driven out on the FSR pin. |
| 0 | 1 | 1 | Internal FSR is driven by the sample rate generator frame-synchronization signal (FSG). | Input. The external frame-synchronization input on the FSR pin is used to synchronize CLKG and generate FSG pulses. |
| 1 | 0 | 0 | Internal FSX drives internal FSR. | High impedance |
| 1 | 0 or 1 | 1 | Internal FSX drives internal FSR. | Input. If the sample rate generator is running, external FSR is used to synchronize CLKG and generate FSG pulses. |
| 1 | 1 | 0 | Internal FSX drives internal FSR. | Output. Receive (same as transmit) frame synchronization is inverted as determined by FSRP before being driven out on the FSR pin. |

### *21.8.16 Receive Frame-Synchronization Polarity*

The FSRP bit (see Table 21-38) determines whether frame-synchronization pulses are active high or active low on the FSR pin.

**Table 21-38. Register Bit Used to Set Receive Frame-Synchronization Polarity**

| Register | Bit | Name | Function | | Type | Reset Value |
|---|---|---|---|---|---|---|
| PCR | 2 | FSRP | Receive frame-synchronization polarity | | R/W | 0 |
| | | | FSRP = 0 | Frame-synchronization pulse FSR is active high. | | |
| | | | FSRP = 1 | Frame-synchronization pulse FSR is active low. | | |



Copyright © 2024 Texas Instruments Incorporated

#### 21.8.16.1 Frame-Synchronization Pulses, Clock Signals, and Their Polarities

Receive frame-synchronization pulses can be generated internally by the sample rate generator (see Section 21.4.2) or driven by an external source. The source of frame synchronization is selected by programming the mode bit, FSRM, in PCR. FSR is also affected by the GSYNC bit in SRGR2. For information about the effects of FSRM and GSYNC, see Section 21.8.15. Similarly, receive clocks can be selected to be inputs or outputs by programming the mode bit, CLKRM, in the PCR (see Section 21.8.17).

When FSR and FSX are inputs (FSXM = FSRM= 0, external frame-synchronization pulses), the McBSP detects them on the internal falling edge of clock, internal MCLKR, and internal CLKX, respectively. The receive data arriving at the DR pin is also sampled on the falling edge of internal MCLKR. These internal clock signals are either derived from an external source via CLK(R/X) pins or driven by the sample rate generator clock (CLKG) internal to the McBSP.

When FSR and FSX are outputs, implying that the outputs are driven by the sample rate generator, the outputs are generated (transition to their active state) on the rising edge of the internal clock, CLK(R/X). Similarly, data on the DX pin is output on the rising edge of internal CLKX.

FSRP, FSXP, CLKRP, and CLKXP in the pin control register (PCR) configure the polarities of the FSR, FSX, MCLKR, and CLKX signals, respectively. All frame-synchronization signals (internal FSR, internal FSX) that are internal to the serial port are active high. If the serial port is configured for external frame synchronization (FSR/FSX are inputs to McBSP), and FSRP = FSXP = 1, the external active-low frame-synchronization signals are inverted before being sent to the receiver (internal FSR) and transmitter (internal FSX). Similarly, if internal synchronization (FSR/FSX are output pins and GSYNC = 0) is selected, the internal active-high frame-synchronization signals are inverted, if the polarity bit FS(R/X)P = 1, before being sent to the FS(R/X) pin.

On the transmit side, the transmit clock polarity bit, CLKXP, sets the edge used to shift and clock out transmit data. Data is always transmitted on the rising edge of internal CLKX. If CLKXP = 1 and external clocking is selected (CLKXM = 0 and CLKX is an input), the external falling-edge triggered input clock on CLKX is inverted to a rising-edge triggered clock before being sent to the transmitter. If CLKXP = 1, and internal clocking selected (CLKXM = 1 and CLKX is an output pin), the internal (rising-edge triggered) clock, internal CLKX, is inverted before being sent out on the MCLKX pin.

Similarly, the receiver can reliably sample data that is clocked with a rising edge clock (by the transmitter). The receive clock polarity bit, CLKRP, sets the edge used to sample received data. The receive data is always sampled on the falling edge of internal MCLKR. Therefore, if CLKRP = 1 and external clocking is selected (CLKRM = 0 and MCLKR is an input pin), the external rising-edge triggered input clock on MCLKR is inverted to a falling-edge triggered clock before being sent to the receiver. If CLKRP = 1 and internal clocking is selected (CLKRM = 1), the internal falling-edge triggered clock is inverted to a rising-edge triggered clock before being sent out on the MCLKR pin.

MCLKRP = CLKXP in a system where the same clock (internal or external) is used to clock the receiver and transmitter. The receiver uses the opposite edge as the transmitter to make sure of valid setup and hold of data around this edge. Figure 21-48 shows how data clocked by an external serial device using a rising edge can be sampled by the McBSP receiver on the falling edge of the same clock.

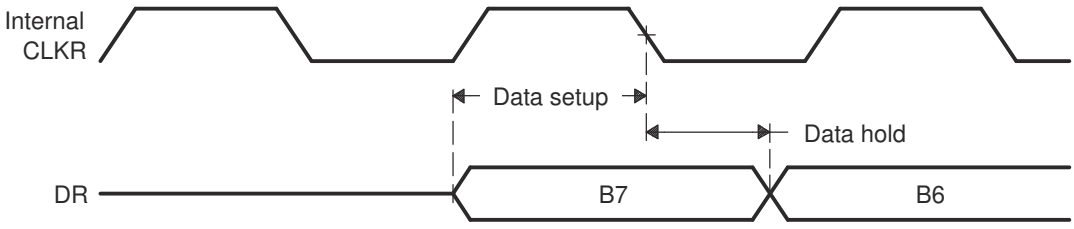



**Figure 21-48. Data Clocked Externally Using a Rising Edge and Sampled by the McBSP Receiver on a Falling Edge**

Set the SRG Frame-Synchronization Period and Pulse Width.



Copyright © 2024 Texas Instruments Incorporated

#### 21.8.16.2 Frame-Synchronization Period and the Frame-Synchronization Pulse Width

The sample rate generator can produce a clock signal, CLKG, and a frame-synchronization signal, FSG. If the sample rate generator is supplying receive or transmit frame synchronization, you must program the bit fields FPER and FWID.

On FSG, the period from the start of a frame-synchronization pulse to the start of the next pulse is (FPER + 1) CLKG cycles. The 12 bits of FPER allow a frame-synchronization period of 1 to 4096 CLKG cycles, which allows up to 4096 data bits per frame. When GSYNC = 1, FPER is a don't care value.

Each pulse on FSG has a width of (FWID + 1) CLKG cycles. The eight bits of FWID allow a pulse width of 1 to 256 CLKG cycles. It is recommended that FWID be programmed to a value less than the programmed word length.

The values in FPER and FWID are loaded into separate down-counters. The 12-bit FPER counter counts down the generated clock cycles from the programmed value (4095 maximum) to 0. The 8-bit FWID counter counts down from the programmed value (255 maximum) to 0. Table 21-39 shows settings for FPER and FWID.

Figure 21-49 shows a frame-synchronization period of 16 CLKG periods (FPER = 15 or 00001111b) and a frame-synchronization pulse with an active width of 2 CLKG periods (FWID = 1).

**Table 21-39. Register Bits Used to Set the SRG Frame-Synchronization Period and Pulse Width**

| Register | Bit | Name | Function | Type | Reset Value |
|---|---|---|---|---|---|
| SRGR2 | 11-0 | FPER | Sample rate generator frame-synchronization period<br>For the frame-synchronization signal FSG, (FPER + 1) determines the period from the start of a frame-synchronization pulse to the start of the next frame-synchronization pulse.<br>Range for (FPER + 1):<br>1 to 4096 CLKG cycles | R/W | 0000 0000 0000 |
| SRGR1 | 15-8 | FWID | Sample rate generator frame-synchronization pulse width<br>This field plus 1 determines the width of each frame-synchronization pulse on FSG.<br>Range for (FWID + 1):<br>1 to 256 CLKG cycles | R/W | 0000 0000 |

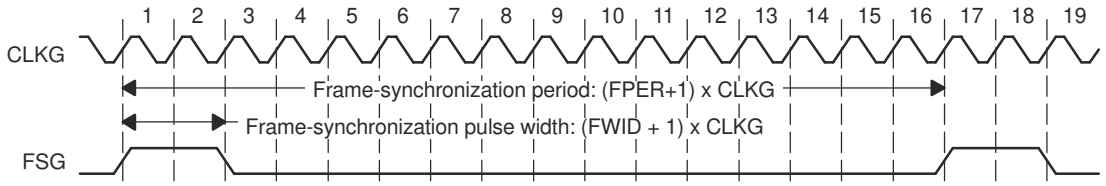



**Figure 21-49. Frame of Period 16 CLKG Periods and Active Width of 2 CLKG Periods**

When the sample rate generator comes out of reset, FSG is in the inactive state. Then, when GRST = 1 and FSGM = 1, a frame-synchronization pulse is generated. The frame width value (FWID + 1) is counted down on every CLKG cycle until the value reaches 0, at which time FSG goes low. At the same time, the frame period value (FPER + 1) is also counting down. When this value reaches 0, FSG goes high, indicating a new frame.



Copyright © 2024 Texas Instruments Incorporated

### 21.8.17 Receive Clock Mode

Table 21-40 shows the settings for bits used to set receive clock mode.

**Table 21-40. Register Bits Used to Set the Receive Clock Mode**

| Register | Bit | Name | Function | | Type | Reset Value |
|---|---|---|---|---|---|---|
| PCR | 8 | CLKRM | Receive clock mode | | R/W | 0 |
| | | | Case 1: Digital loopback mode not set (DLB = 0) in SPCR1. | | | |
| | | | CLKRM = 0 | The MCLKR pin is an input pin that supplies the internal receive clock (MCLKR). | | |
| | | | CLKRM = 1 | Internal MCLKR is driven by the sample rate generator of the McBSP. The MCLKR pin is an output pin that reflects internal MCLKR. | | |
| | | | Case 2: Digital loopback mode set (DLB = 1) in SPCR1. | | | |
| | | | CLKRM = 0 | The MCLKR pin is in the high impedance state. The internal receive clock (MCLKR) is driven by the internal transmit clock (CLKX). Internal CLKX is derived according to the CLKXM bit of PCR. | | |
| | | | CLKRM = 1 | Internal MCLKR is driven by internal CLKX. The MCLKR pin is an output pin that reflects internal MCLKR. Internal CLKX is derived according to the CLKXM bit of PCR. | | |
| SPCR1 | 15 | DLB | Digital loopback mode | | R/W | 00 |
| | | | DLB = 0 | Digital loopback mode is disabled. | | |
| | | | DLB = 1 | Digital loopback mode is enabled. The receive signals, including the receive frame-synchronization signal, are connected internally through multiplexers to the corresponding transmit signals. | | |
| SPCR1 | 12-11 | CLKSTP | Clock stop mode | | R/W | 00 |
| | | | CLKSTP = 0Xb | Clock stop mode disabled; normal clocking for non-SPI mode. | | |
| | | | CLKSTP = 10b | Clock stop mode enabled without clock delay. The internal receive clock signal (MCLKR) and the internal receive frame-synchronization signal (FSR) are internally connected to their transmit counterparts, CLKX and FSX. | | |
| | | | CLKSTP = 11b | Clock stop mode enabled with clock delay. The internal receive clock signal (MCLKR) and the internal receive frame-synchronization signal (FSR) are internally connected to their transmit counterparts, CLKX and FSX. | | |



Copyright © 2024 Texas Instruments Incorporated

#### 21.8.17.1 Selecting a Source for the Receive Clock and a Data Direction for the MCLKR Pin

Table 21-41 shows how you can select various sources to provide the receive clock signal and affect the MCLKR pin. The polarity of the signal on the MCLKR pin is determined by the CLKRP bit.

In the digital loopback mode (DLB = 1), the transmit clock signal is used as the receive clock signal.

Also, in the clock stop mode, the internal receive clock signal (MCLKR) and the internal receive frame-synchronization signal (FSR) are internally connected to their transmit counterparts, CLKX and FSX.

**Table 21-41. Receive Clock Signal Source Selection**

| DLB in SPCR1 | CLKRM in PCR | Source of Receive Clock | MCLKR Pin Status |
|---|---|---|---|
| 0 | 0 | The MCLKR pin is an input driven by an external clock. The external clock signal is inverted as determined by CLKRP before being used. | Input |
| 0 | 1 | The sample rate generator clock (CLKG) drives internal MCLKR. | Output. CLKG, inverted as determined by CLKRP, is driven out on the MCLKR pin. |
| 1 | 0 | Internal CLKX drives internal MCLKR. To configure CLKX, see Section 21.9.19. | High impedance |
| 1 | 1 | Internal CLKX drives internal MCLKR. To configure CLKX, see Section 21.9.19. | Output. Internal MCLKR (same as internal CLKX) is inverted as determined by CLKRP before being driven out on the MCLKR pin. |

### *21.8.18 Receive Clock Polarity*

Table 21-42 shows which register bits set the Receive Clock Polarity.

**Table 21-42. Register Bit Used to Set Receive Clock Polarity**

| Register | Bit | Name | Function | | Type | Reset Value |
|---|---|---|---|---|---|---|
| PCR | 0 | CLKRP | Receive clock polarity | | R/W | 0 |
| | | | CLKRP = 0 | Receive data sampled on falling edge of MCLKR | | |
| | | | CLKRP = 1 | Receive data sampled on rising edge of MCLKR | | |



Copyright © 2024 Texas Instruments Incorporated

#### 21.8.18.1 Frame Synchronization Pulses, Clock Signals, and Their Polarities

Receive frame-synchronization pulses can be generated internally by the sample rate generator (see Section 21.4.2) or driven by an external source. The source of frame synchronization is selected by programming the mode bit, FSRM, in PCR. FSR is also affected by the GSYNC bit in SRGR2. For information about the effects of FSRM and GSYNC, see Section 21.8.15. Similarly, receive clocks can be selected to be inputs or outputs by programming the mode bit, CLKRM, in the PCR (see Section 21.8.17).

When FSR and FSX are inputs (FSXM = FSRM= 0, external frame-synchronization pulses), the McBSP detects them on the internal falling edge of clock, internal MCLKR, and internal CLKX, respectively. The receive data arriving at the DR pin is also sampled on the falling edge of internal MCLKR. These internal clock signals are either derived from external source via CLK(R/X) pins or driven by the sample rate generator clock (CLKG) internal to the McBSP.

When FSR and FSX are outputs, implying that the outputs are driven by the sample rate generator, the outputs are generated (transition to their active state) on the rising edge of internal clock, CLK(R/X). Similarly, data on the DX pin is output on the rising edge of internal CLKX.

FSRP, FSXP, CLKRP, and CLKXP in the pin control register (PCR) configure the polarities of the FSR, FSX, MCLKR, and CLKX signals, respectively. All frame-synchronization signals (internal FSR, internal FSX) that are internal to the serial port are active high. If the serial port is configured for external frame synchronization (FSR/FSX are inputs to McBSP) and FSRP = FSXP = 1, the external active-low frame-synchronization signals are inverted before being sent to the receiver (internal FSR) and transmitter (internal FSX). Similarly, if internal synchronization (FSR/FSX are output pins and GSYNC = 0) is selected, the internal active-high frame-synchronization signals are inverted, if the polarity bit FS(R/X)P = 1, before being sent to the FS(R/X) pin.

On the transmit side, the transmit clock polarity bit, CLKXP, sets the edge used to shift and clock out transmit data. Data is always transmitted on the rising edge of internal CLKX. If CLKXP = 1 and external clocking is selected (CLKXM = 0 and CLKX is an input), the external falling-edge triggered input clock on CLKX is inverted to a rising-edge triggered clock before being sent to the transmitter. If CLKXP = 1 and internal clocking is selected (CLKXM = 1 and CLKX is an output pin), the internal (rising-edge triggered) clock, internal CLKX, is inverted before being sent out on the MCLKX pin.

Similarly, the receiver can reliably sample data that is clocked with a rising edge clock (by the transmitter). The receive clock polarity bit, CLKRP, sets the edge used to sample received data. The receive data is always sampled on the falling edge of internal MCLKR. Therefore, if CLKRP = 1 and external clocking is selected (CLKRM = 0 and MCLKR is an input pin), the external rising-edge triggered input clock on MCLKR is inverted to a falling-edge triggered clock before being sent to the receiver. If CLKRP = 1 and internal clocking is selected (CLKRM = 1), the internal falling-edge triggered clock is inverted to a rising-edge triggered clock before being sent out on the MCLKR pin.

CLKRP = CLKXP in a system where the same clock (internal or external) is used to clock the receiver and transmitter. The receiver uses the opposite edge as the transmitter to make sure of valid setup and hold of data around this edge. Figure 21-50 shows how data clocked by an external serial device using a rising edge can be sampled by the McBSP receiver on the falling edge of the same clock.

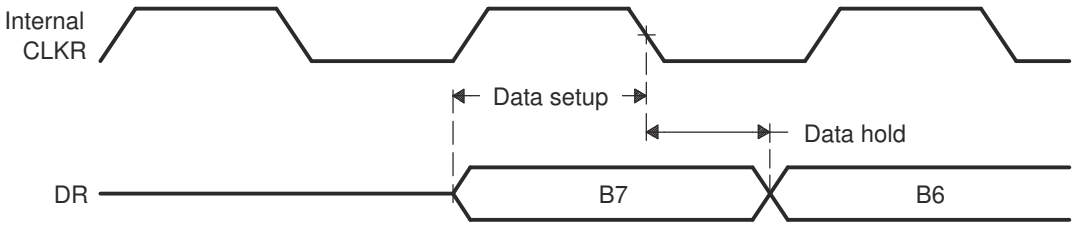



**Figure 21-50. Data Clocked Externally Using a Rising Edge and Sampled by the McBSP Receiver on a Falling Edge**



Copyright © 2024 Texas Instruments Incorporated

### 21.8.19 SRG Clock Divide-Down Value

**Table 21-43. Register Bits Used to Set the Sample Rate Generator (SRG) Clock Divide-Down Value**

| Register | Bit | Name | Function | Type | Reset Value |
|---|---|---|---|---|---|
| SRGR1 | 7-0 | CLKGDV | Sample rate generator clock divide-down value<br>The input clock of the sample rate generator is divided by (CLKGDV + 1) to generate the required sample rate generator clock frequency. The default value of CLKGDV is 1 (divide input clock by 2). | R/W | 0000 0001 |

#### 21.8.19.1 Sample Rate Generator Clock Divider

The first divider stage generates the serial data bit clock from the input clock. This divider stage utilizes a counter, preloaded by CLKGDV, that contains the divide ratio value.

The output of the first divider stage is the data bit clock, which is output as CLKG and which serves as the input for the second and third stages of the divider.

CLKG has a frequency equal to 1/(CLKGDV + 1) of sample rate generator input clock. Thus, the sample generator input clock frequency is divided by a value between 1 and 256. When CLKGDV is odd or equal to 0, the CLKG duty cycle is 50%. When CLKGDV is an even value, 2p, representing an odd divide-down, the high-state duration is p + 1 cycles and the low-state duration is p cycles.

### 21.8.20 SRG Clock Synchronization Mode

For more details on using the clock synchronization feature, see Section 21.4.3.

**Table 21-44. Register Bit Used to Set the SRG Clock Synchronization Mode**

| Register | Bit | Name | Function | Type | Reset Value |
|---|---|---|---|---|---|
| SRGR2 | 15 | GSYNC | Sample rate generator clock synchronization<br>GSYNC is used only when the input clock source for the sample rate generator is external—on the MCLKR or MCLKX pin.<br>GSYNC = 0: The sample rate generator clock (CLKG) is free running. CLKG oscillates without adjustment, and FSG pulses every (FPER + 1) CLKG cycles.<br>GSYNC = 1: Clock synchronization is performed. When a pulse is detected on the FSR pin:<br>• CLKG is adjusted as necessary so that CLKG is synchronized with the input clock on the MCLKR or MCLKX pin.<br>• FSG pulses. FSG only pulses in response to a pulse on the FSR pin. The frame-synchronization period defined in FPER is ignored. | R/W | 0 |



Copyright © 2024 Texas Instruments Incorporated

### 21.8.21 SRG Clock Mode (Choose an Input Clock)

The sample rate generator can produce a clock signal (CLKG) for use by the receiver, the transmitter, or both, but CLKG is derived from an input clock. Table 21-45 shows the four possible sources of the input clock. For more details on generating CLKG, see Section 21.4.1.1.

**Table 21-45. Register Bits Used to Set the SRG Clock Mode (Choose an Input Clock)**

| Register | Bit | Name | Function | | Type | Reset Value |
|---|---|---|---|---|---|---|
| PCR | 7 | SCLKME | Sample rate generator clock mode | | R/W | 0 |
| SRGR2 | 13 | CLKSM | | | R/W | 1 |
| | | | SCLKME = 0<br>CLKSM = 0 | Reserved | | |
| | | | SCLKME = 0<br>CLKSM = 1 | Sample rate generator clock derived from LSPCLK (default) | | |
| | | | SCLKME = 1<br>CLKSM = 0 | Sample rate generator clock derived from MCLKR pin | | |
| | | | SCLKME = 1<br>CLKSM = 1 | Sample rate generator clock derived from MCLKX pin | | |

### 21.8.22 SRG Input Clock Polarity

Table 21-46 shows which register bits set the SRG Input Clock Polarity.

**Table 21-46. Register Bits Used to Set the SRG Input Clock Polarity**

| Register | Bit | Name | Function | | Type | Reset Value |
|---|---|---|---|---|---|---|
| PCR | 1 | CLKXP | MCLKX pin polarity | | R/W | 0 |
| | | | CLKXP determines the input clock polarity when the MCLKX pin supplies the input clock (SCLKME = 1 and CLKSM = 1). | | | |
| | | | CLKXP = 0 | Rising edge on MCLKX pin generates transitions on CLKG and FSG. | | |
| | | | CLKXP = 1 | Falling edge on MCLKX pin generates transitions on CLKG and FSG. | | |
| PCR | 0 | CLKRP | MCLKR pin polarity | | R/W | 0 |
| | | | CLKRP determines the input clock polarity when the MCLKR pin supplies the input clock (SCLKME = 1 and CLKSM = 0). | | | |
| | | | CLKRP = 0 | Falling edge on MCLKR pin generates transitions on CLKG and FSG. | | |
| | | | CLKRP = 1 | Rising edge on MCLKR pin generates transitions on CLKG and FSG. | | |



Copyright © 2024 Texas Instruments Incorporated

#### 21.8.22.1 Using CLKXP/CLKRP to Choose an Input Clock Polarity

The sample rate generator can produce a clock signal (CLKG) and a frame-synchronization signal (FSG) for use by the receiver, the transmitter, or both. To produce CLKG and FSG, the sample rate generator must be driven by an input clock signal derived from the CPU clock or from an external clock on the CLKX or MCLKR pin. If you use a pin, choose a polarity for that pin by using the appropriate polarity bit (CLKXP for the MCLKX pin, CLKRP for the MCLKR pin). The polarity determines whether the rising or falling edge of the input clock generates transitions on CLKG and FSG.

## 21.9 Transmitter Configuration

To configure the McBSP transmitter, perform the following procedure:

1. Place the McBSP/transmitter in reset (see Section 21.9.2).
2. Program the McBSP registers for the desired transmitter operation (see Section 21.9.1).
3. Take the transmitter out of reset (see Section 21.9.2).

### *21.9.1 Programming the McBSP Registers for the Desired Transmitter Operation*

The following is a list of important tasks to be performed when you are configuring the McBSP transmitter. Each task corresponds to one or more McBSP register bit fields.

- Global behavior:
  - Set the transmitter pins to operate as McBSP pins.
  - Enable/disable the digital loopback mode.
  - Enable/disable the clock stop mode.
  - Enable/disable transmit multichannel selection.
- Data behavior:
  - Choose 1 or 2 phases for the transmit frame.
  - Set the transmit word length(s).
  - Set the transmit frame length.
  - Enable/disable the transmit frame-synchronization ignore function.
  - Set the transmit companding mode.
  - Set the transmit data delay.
  - Set the transmit DXENA mode.
  - Set the transmit interrupt mode.
- Frame-synchronization behavior:
  - Set the transmit frame-synchronization mode.
  - Set the transmit frame-synchronization polarity.
  - Set the SRG frame-synchronization period and pulse width.
- Clock behavior:
  - Set the transmit clock mode.
  - Set the transmit clock polarity.
  - Set the SRG clock divide-down value.
  - Set the SRG clock synchronization mode.
  - Set the SRG clock mode (choose an input clock).
  - Set the SRG input clock polarity.



Copyright © 2024 Texas Instruments Incorporated

### 21.9.2 Resetting and Enabling the Transmitter

The first step of the transmitter configuration procedure is to reset the transmitter, and the last step is to enable the transmitter (to take the transmitter out of reset). Table 21-47 describes the bits used for both of these steps.

**Table 21-47. Register Bits Used to Place Transmitter in Reset Field Descriptions**

| Register | Bit | Field | Value | Description |
|---|---|---|---|---|
| SPCR2 | 7 | FRST | | Frame-synchronization logic reset |
| | | | 0 | Frame-synchronization logic is reset. The sample rate generator does not generate frame-synchronization signal FSG, even if GRST = 1. |
| | | | 1 | Frame-synchronization is enabled. If GRST = 1, frame-synchronization signal FSG is generated after (FPER + 1) number of CLKG clock cycles; all frame counters are loaded with their programmed values. |
| SPCR2 | 6 | GRST | | Sample rate generator reset |
| | | | 0 | Sample rate generator is reset. If GRST = 0 due to a device reset, CLKG is driven by the CPU clock divided by 2, and FSG is driven low (inactive). If GRST = 0 due to program code, CLKG and FSG are both driven low (inactive). |
| | | | 1 | Sample rate generator is enabled. CLKG is driven according to the configuration programmed in the sample rate generator registers (SRGR[1,2]). If FRST = 1, the generator also generates the frame-synchronization signal FSG as programmed in the sample rate generator registers. |
| SPCR2 | 0 | XRST | | Transmitter reset |
| | | | 0 | The serial port transmitter is disabled and in the reset state. |
| | | | 1 | The serial port transmitter is enabled. |

#### 21.9.2.1 Reset Considerations

The serial port can be reset in the following two ways:

1. A DSP reset ( $\overline{\text{XRS}}$ signal driven low) places the receiver, transmitter, and sample rate generator in reset. When the device reset is removed, GRST = FRST = RRST = XRST = 0, keeping the entire serial port in the reset state.
2. The serial port transmitter and receiver can be reset directly using the RRST and XRST bits in the serial port control registers. The sample rate generator can be reset directly using the GRST bit in SPCR2.
3. When using the DMA, the order in which McBSP events must occur is important. DMA channel and peripheral interrupts must be configured prior to releasing the McBSP transmitter from reset.

   The reason for this is that an XRDY is fired when XRST = 1. The XRDY signals the DMA to start copying data from the buffer into the transmit register. If the McBSP transmitter is released from reset before the DMA channel and peripheral interrupts are configured, the XRDY signals before the DMA channel can receive the signal; therefore, the DMA does not move the data from the buffer to the transmit register. The DMA PERINTFLG is edge-sensitive and will fail to recognize the XRDY, which is continuously high.

For more details about McBSP reset conditions and effects, see Section 21.10.2.

### 21.9.3 Set the Transmitter Pins to Operate as McBSP Pins

To configure a pin for McBSP functioning, configure the bits of the GPxMUXn register appropriately. In addition to this, bits 12 and 13 of the PCR register must be set to 0. These bits are defined as reserved.



Copyright © 2024 Texas Instruments Incorporated

### 21.9.4 Digital Loopback Mode

The DLB bit determines whether the digital loopback mode is on. DLB is described in Table 21-48.

**Table 21-48. Register Bit Used to Enable/Disable the Digital Loopback Mode**

| Register | Bit | Name | Function | | Type | Reset Value |
|---|---|---|---|---|---|---|
| SPCR1 | 15 | DLB | Digital loopback mode | | R/W | 0 |
| | | | DLB = 0 | Digital loopback mode is disabled. | | |
| | | | DLB = 1 | Digital loopback mode is enabled. | | |

In the digital loopback mode, the receive signals are connected internally through multiplexers to the corresponding transmit signals, as shown in Table 21-49. This mode allows testing of serial port code with a single DSP device; the McBSP receives the data it transmits.

**Table 21-49. Receive Signals Connected to Transmit Signals in Digital Loopback Mode**

| This Receive Signal | Is Fed Internally by This Transmit Signal |
|---|---|
| DR (receive data) | DX (transmit data) |
| FSR (receive frame synchronization) | FSX (transmit frame synchronization) |
| MCLKR (receive clock) | CLKX (transmit clock) |

### 21.9.5 Clock Stop Mode

The CLKSTP bits determine whether the clock stop mode is on. CLKSTP is described in Table 21-50.

**Table 21-50. Register Bits Used to Enable/Disable the Clock Stop Mode**

| Register | Bit | Name | Function | | Type | Reset Value |
|---|---|---|---|---|---|---|
| SPCR1 | 12-11 | CLKSTP | Clock stop mode | | R/W | 00 |
| | | | CLKSTP = 0Xb | Clock stop mode disabled; normal clocking for non-SPI mode. | | |
| | | | CLKSTP = 10b | Clock stop mode enabled without clock delay | | |
| | | | CLKSTP = 11b | Clock stop mode enabled with clock delay | | |

The clock stop mode supports the SPI master-slave protocol. If you do not plan to use the SPI protocol, you can clear CLKSTP to disable the clock stop mode.

In the clock stop mode, the clock stops at the end of each data transfer. At the beginning of each data transfer, the clock starts immediately (CLKSTP = 10b) or after a half-cycle delay (CLKSTP = 11b). The CLKXP bit determines whether the starting edge of the clock on the MCLKX pin is rising or falling. The CLKRP bit determines whether receive data is sampled on the rising or falling edge of the clock shown on the MCLKR pin.

Table 21-51 summarizes the impact of CLKSTP, CLKXP, and CLKRP on serial port operation. In the clock stop mode, the receive clock is tied internally to the transmit clock, and the receive frame-synchronization signal is tied internally to the transmit frame-synchronization signal.



Copyright © 2024 Texas Instruments Incorporated

**Table 21-51. Effects of CLKSTP, CLKXP, and CLKRP on the Clock Scheme**

| Bit Settings | Clock Scheme |
|---|---|
| CLKSTP = 00b or 01b<br>CLKXP = 0 or 1<br>CLKRP = 0 or 1 | Clock stop mode disabled. Clock enabled for non-SPI mode. |
| CLKSTP = 10b<br>CLKXP = 0<br>CLKRP = 0 | Low inactive state without delay: The McBSP transmits data on the rising edge of CLKX and receives data on the falling edge of MCLKR. |
| CLKSTP = 11b<br>CLKXP = 0<br>CLKRP = 1 | Low inactive state with delay: The McBSP transmits data one-half cycle ahead of the rising edge of CLKX and receives data on the rising edge of MCLKR. |
| CLKSTP = 10b<br>CLKXP = 1<br>CLKRP = 0 | High inactive state without delay: The McBSP transmits data on the falling edge of CLKX and receives data on the rising edge of MCLKR. |
| CLKSTP = 11b<br>CLKXP = 1<br>CLKRP = 1 | High inactive state with delay: The McBSP transmits data one-half cycle ahead of the falling edge of CLKX and receives data on the falling edge of MCLKR. |

### *21.9.6 Transmit Multichannel Selection Mode*

For more details, see Section 21.6.7.

**Table 21-52. Register Bits Used to Enable/Disable Transmit Multichannel Selection**

| Register | Bit | Name | Function | | Type | Reset Value |
|---|---|---|---|---|---|---|
| MCR2 | 1-0 | XMCM | Transmit multichannel selection | | R/W | 00 |
| | | | XMCM = 00b | No transmit multichannel selection mode is on. All channels are enabled and unmasked. No channels can be disabled or masked. | | |
| | | | XMCM = 01b | All channels are disabled unless the channels are selected in the appropriate transmit channel enable registers (XCERs). If enabled, a channel in this mode is also unmasked.<br>The XMCME bit determines whether 32 channels or 128 channels are selectable in XCERs. | | |
| | | | XMCM = 10b | All channels are enabled, but the channels are masked unless the channels are selected in the appropriate transmit channel enable registers (XCERs).<br>The XMCME bit determines whether 32 channels or 128 channels are selectable in XCERs. | | |
| | | | XMCM = 11b | This mode is used for symmetric transmission and reception.<br>All channels are disabled for transmission unless the channels are enabled for reception in the appropriate receive channel enable registers (RCERs). Once enabled, the channels are masked unless the channels are also selected in the appropriate transmit channel enable registers (XCERs).<br>The XMCME bit determines whether 32 channels or 128 channels are selectable in RCERs and XCERs. | | |



Copyright © 2024 Texas Instruments Incorporated

### 21.9.7 XCERs Used in the Transmit Multichannel Selection Mode

For multichannel selection operation, the assignment of channels to the XCERs depends on whether 32 or 128 channels are individually selectable by the XMCME bit, as shown in Table 21-53. The table shows which block of channels is assigned to each XCER that is used. For each XCER, the table shows which channel is assigned to each of the bits.

**Note**

When XMCM = 11b (for symmetric transmission and reception), the transmitter uses the receive channel enable registers (RCERs) to enable channels and uses the XCERs to unmask channels for transmission.

**Table 21-53. Use of the Transmit Channel Enable Registers**

| Number of Selectable Channels | Block Assignments | | Channel Assignments | |
|---|---|---|---|---|
| | XCERx | Block Assigned | Bit in XCERx | Channel Assigned |
| 32 (XMCME = 0) | XCERA | Channels n to (n + 15) | XCE0 | Channel n |
| | | | XCE1 | Channel (n + 1) |
| | | | XCE2 | Channel (n + 2) |
| | | | : | : |
| | | When XMCM = 01b or 10b, the block of channels is chosen with the XPABLK bits. When XMCM = 11b, the block is chosen with the RPABLK bits. | XCE15 | Channel (n + 15) |
| | XCERB | Channels m to (m + 15) | XCE0 | Channel m |
| | | | XCE1 | Channel (m + 1) |
| | | | XCE2 | Channel (m + 2) |
| | | | : | : |
| | | When XMCM = 01b or 10b, the block of channels is chosen with the XPBBLK bits. When XMCM = 11b, the block is chosen with the RPBBLK bits. | XCE15 | Channel (m + 15) |
| 128 (XMCME = 1) | XCERA | Block 0 | XCE0 | Channel 0 |
| | | | XCE1 | Channel 1 |
| | | | XCE2 | Channel 2 |
| | | | : | : |
| | | | XCE15 | Channel 15 |
| | XCERB | Block 1 | XCE0 | Channel 16 |
| | | | XCE1 | Channel 17 |
| | | | XCE2 | Channel 18 |
| | | | : | : |
| | | | XCE15 | Channel 31 |
| | XCERC | Block 2 | XCE0 | Channel 32 |
| | | | XCE1 | Channel 33 |
| | | | XCE2 | Channel 34 |
| | | | : | : |
| | | | XCE15 | Channel 47 |
| | XCERD | Block 3 | XCE0 | Channel 48 |
| | | | XCE1 | Channel 49 |
| | | | XCE2 | Channel 50 |
| | | | : | : |
| | | | XCE15 | Channel 63 |



Copyright © 2024 Texas Instruments Incorporated

**Table 21-53. Use of the Transmit Channel Enable Registers (continued)**

| Number of Selectable Channels | Block Assignments | | Channel Assignments | |
|---|---|---|---|---|
| | XCERx | Block Assigned | Bit in XCERx | Channel Assigned |
| | XCERE | Block 4 | XCE0 | Channel 64 |
| | | | XCE1 | Channel 65 |
| | | | XCE2 | Channel 66 |
| | | | : | : |
| | | | XCE15 | Channel 79 |
| | XCERF | Block 5 | XCE0 | Channel 80 |
| | | | XCE1 | Channel 81 |
| | | | XCE2 | Channel 82 |
| | | | : | : |
| | | | XCE15 | Channel 95 |
| | XCERG | Block 6 | XCE0 | Channel 96 |
| | | | XCE1 | Channel 97 |
| | | | XCE2 | Channel 98 |
| | | | : | : |
| | | | XCE15 | Channel 111 |
| | XCERH | Block 7 | XCE0 | Channel 112 |
| | | | XCE1 | Channel 113 |
| | | | XCE2 | Channel 114 |
| | | | : | : |
| | | | XCE15 | Channel 127 |



Copyright © 2024 Texas Instruments Incorporated

### 21.9.8 Transmit Frame Phases

The XPHASE bit (see Table 21-54) determines whether the transmit data frame has one or two phases.

**Table 21-54. Register Bit Used to Choose 1 or 2 Phases for the Transmit Frame**

| Register | Bit | Name | Function | | Type | Reset Value |
|---|---|---|---|---|---|---|
| XCR2 | 15 | XPHASE | Transmit phase number | | R/W | 0 |
| | | | Specifies whether the transmit frame has 1 or 2 phases. | | | |
| | | | XPHASE = 0 | Single-phase frame | | |
| | | | XPHASE = 1 | Dual-phase frame | | |

### 21.9.9 Transmit Word Lengths

The XWDLEN1 and XWDLEN2 bit fields (see Table 21-55) determine how many bits are in each serial word in phase 1 and in phase 2, respectively, of the transmit data frame.

**Table 21-55. Register Bits Used to Set the Transmit Word Lengths**

| Register | Bit | Name | Function | | Type | Reset Value |
|---|---|---|---|---|---|---|
| XCR1 | 7-5 | XWDLEN1 | Transmit word length of frame phase 1 | | R/W | 000 |
| | | | XWDLEN1 = 000b | 8 bits | | |
| | | | XWDLEN1 = 001b | 12 bits | | |
| | | | XWDLEN1 = 010b | 16 bits | | |
| | | | XWDLEN1 = 011b | 20 bits | | |
| | | | XWDLEN1 = 100b | 24 bits | | |
| | | | XWDLEN1 = 101b | 32 bits | | |
| | | | XWDLEN1 = 11Xb | Reserved | | |
| XCR2 | 7-5 | XWDLEN2 | Transmit word length of frame phase 2 | | R/W | 000 |
| | | | XWDLEN2 = 000b | 8 bits | | |
| | | | XWDLEN2 = 001b | 12 bits | | |
| | | | XWDLEN2 = 010b | 16 bits | | |
| | | | XWDLEN2 = 011b | 20 bits | | |
| | | | XWDLEN2 = 100b | 24 bits | | |
| | | | XWDLEN2 = 101b | 32 bits | | |
| | | | XWDLEN2 = 11Xb | Reserved | | |

#### 21.9.9.1 Word Length Bits

Each frame can have one or two phases, depending on the value that you load into the RPHASE bit. If a single-phase frame is selected, XWDLEN1 selects the length for every serial word transmitted in the frame. If a dual-phase frame is selected, XWDLEN1 determines the length of the serial words in phase 1 of the frame, and XWDLEN2 determines the word length in phase 2 of the frame.



Copyright © 2024 Texas Instruments Incorporated

### 21.9.10 Transmit Frame Length

The XFRLEN1 and XFRLEN2 bit fields (see Table 21-56) determine how many serial words are in phase 1 and in phase 2, respectively, of the transmit data frame.

**Table 21-56. Register Bits Used to Set the Transmit Frame Length**

| Register | Bit | Name | Function | | Type | Reset Value |
|---|---|---|---|---|---|---|
| XCR1 | 14-8 | XFRLEN1 | Transmit frame length 1 | | R/W | 000 0000 |
| | | | (XFRLEN1 + 1) is the number of serial words in phase 1 of the transmit frame. | | | |
| | | | XFRLEN1 = 000 0000 | 1 word in phase 1 | | |
| | | | XFRLEN1 = 000 0001 | 2 words in phase 1 | | |
| | | | \| | \| | | |
| | | | \| | \| | | |
| | | | XFRLEN1 = 111 1111 | 128 words in phase 1 | | |
| XCR2 | 14-8 | XFRLEN2 | Transmit frame length 2 | | R/W | 000 0000 |
| | | | If a dual-phase frame is selected, (XFRLEN2 + 1) is the number of serial words in phase 2 of the transmit frame. | | | |
| | | | XFRLEN2 = 000 0000 | 1 word in phase 2 | | |
| | | | XFRLEN2 = 000 0001 | 2 words in phase 2 | | |
| | | | \| | \| | | |
| | | | \| | \| | | |
| | | | XFRLEN2 = 111 1111 | 128 words in phase 2 | | |

#### 21.9.10.1 Selected Frame Length

The transmit frame length is the number of serial words in the transmit frame. Each frame can have one or two phases, depending on the value that you load into the XPHASE bit.

If a single-phase frame is selected (XPHASE = 0), the frame length is equal to the length of phase 1. If a dual-phase frame is selected (XPHASE = 1), the frame length is the length of phase 1 plus the length of phase 2.

The 7-bit XFRLEN fields allow up to 128 words per phase. See Table 21-57 for a summary of how to calculate the frame length. This length corresponds to the number of words or logical time slots or channels per frame-synchronization pulse.

**Note**

Program the XFRLEN fields with [*w minus 1*], where w represents the number of words per phase. For example, if you want a phase length of 128 words in phase 1, load 127 into XFRLEN1.

**Table 21-57. How to Calculate Frame Length**

| XPHASE | XFRLEN1 | XFRLEN2 | Frame Length |
|---|---|---|---|
| 0 | 0 ≤ XFRLEN1 ≤ 127 | Don't care | (XFRLEN1 + 1) words |
| 1 | 0 ≤ XFRLEN1 ≤ 127 | 0 ≤ XFRLEN2 ≤ 127 | (XFRLEN1 + 1) + (XFRLEN2 + 1) words |



Copyright © 2024 Texas Instruments Incorporated

### 21.9.11 Enable/Disable the Transmit Frame-Synchronization Ignore Function

**Table 21-58. Register Bit Used to Enable/Disable the Transmit Frame-Synchronization Ignore Function**

| Register | Bit | Name | Function | | Type | Reset Value |
|---|---|---|---|---|---|---|
| XCR2 | 2 | XFIG | Transmit frame-synchronization ignore | | R/W | 0 |
| | | | XFIG = 0 | An unexpected transmit frame-synchronization pulse causes the McBSP to restart the frame transfer. | | |
| | | | XFIG = 1 | The McBSP ignores unexpected transmit frame-synchronization pulses. | | |

#### 21.9.11.1 Unexpected Frame-Synchronization Pulses and Frame-Synchronization Ignore

If a frame-synchronization pulse starts the transfer of a new frame before the current frame is fully transmitted, this pulse is treated as an unexpected frame-synchronization pulse.

When XFIG = 1, normal transmission continues with unexpected frame-synchronization signals ignored.

When XFIG = 0 and an unexpected frame-synchronization pulse occurs, the serial port:

1. Aborts the present transmission
2. Sets XSYNCERR to 1 in SPCR2
3. Reinitiates transmission of the current word that was aborted

For more details about the frame-synchronization error condition, see Section 21.5.6.

#### 21.9.11.2 Examples Showing the Effects of XFIG

Figure 21-51 shows an example in which word B is interrupted by an unexpected frame-synchronization pulse when (R/X)FIG = 0. In the case of transmission, the transmission of B is aborted (B is lost). This condition is a transmit synchronization error, which sets the XSYNCERR bit. No new data has been written to DXR[1,2]; therefore, the McBSP transmits B again.

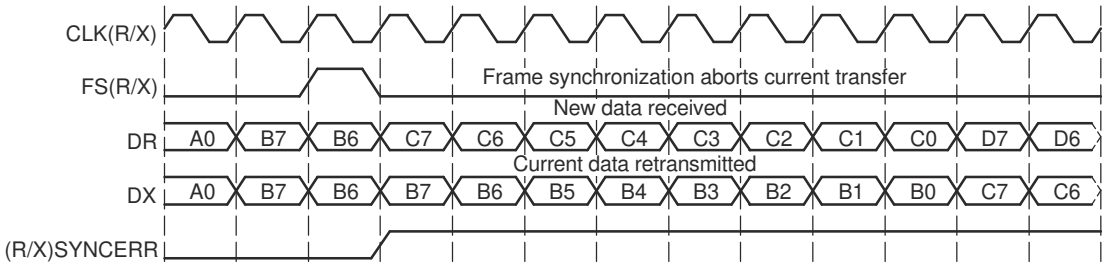



**Figure 21-51. Unexpected Frame-Synchronization Pulse With (R/X) FIG = 0**

In contrast with Figure 21-51, Figure 21-52 shows McBSP operation when unexpected frame-synchronization signals are ignored (when (R/X)FIG = 1). Here, the transfer of word B is not affected by an unexpected frame-synchronization pulse.

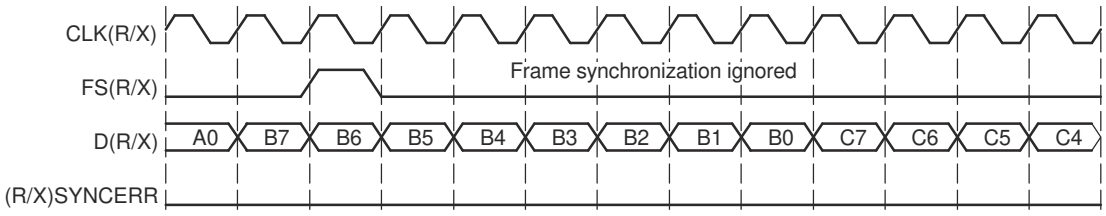



**Figure 21-52. Unexpected Frame-Synchronization Pulse With (R/X) FIG = 1**



Copyright © 2024 Texas Instruments Incorporated

### 21.9.12 Transmit Companding Mode

The XCOMPAND bits (see Table 21-59) determine whether companding or another data transfer option is chosen for McBSP transmission.

**Table 21-59. Register Bits Used to Set the Transmit Companding Mode**

| Register | Bit | Name | Function | | Type | Reset Value |
|---|---|---|---|---|---|---|
| XCR2 | 4-3 | XCOMPAND | Transmit companding mode | | R/W | 00 |
| | | | Modes other than 00b are enabled only when the appropriate XWDLEN is 000b, indicating 8-bit data. | | | |
| | | | XCOMPAND = 00b | No companding, any size data, MSB transmitted first | | |
| | | | XCOMPAND = 01b | No companding, 8-bit data, LSB transmitted first (for details, see Section 21.8.11.4) | | |
| | | | XCOMPAND = 10b | µ-law companding, 8-bit data, MSB transmitted first | | |
| | | | XCOMPAND = 11b | A-law companding, 8-bit data, MSB transmitted first | | |

#### 21.9.12.1 Companding

Companding (COMpressing and exPANDing) hardware allows compression and expansion of data in either µ-law or A-law format. The companding standard employed in the United States and Japan is µ-law. The European companding standard is referred to as A-law. The specifications for µ-law and A-law log PCM are part of the CCITT G.711 recommendation.

A-law and µ-law allow 13 bits and 14 bits of dynamic range, respectively. Any values outside this range are set to the most positive or most negative value. Thus, for companding to work best, the data transferred to and from the McBSP via the CPU or DMA controller must be at least 16 bits wide.

The µ-law and A-law formats both encode data into 8-bit code words. Companded data is always 8 bits wide; the appropriate word length bits (RWDLEN1, RWDLEN2, XWDLEN1, XWDLEN2) must therefore be set to 0, indicating an 8-bit wide serial data stream. If companding is enabled and either of the frame phases does not have an 8-bit word length, companding continues as if the word length is 8 bits.

Figure 21-53 illustrates the companding processes. When companding is chosen for the transmitter, compression occurs during the process of copying data from DXR1 to XSR1. The transmit data is encoded according to the specified companding law (A-law or µ-law). When companding is chosen for the receiver, expansion occurs during the process of copying data from RBR1 to DRR1. The receive data is decoded to twos-complement format.

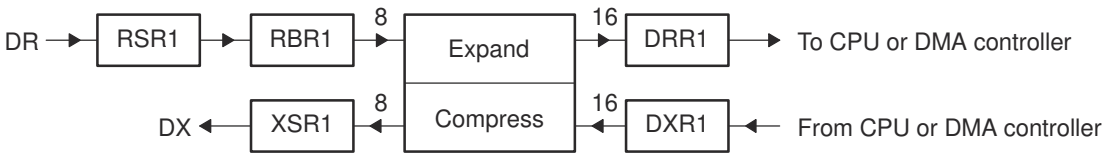



**Figure 21-53. Companding Processes for Reception and for Transmission**



Copyright © 2024 Texas Instruments Incorporated

#### 21.9.12.2 Format for Data To Be Compressed

For transmission using µ-law compression, make sure the 14 data bits are left-justified in DXR1, with the remaining two low-order bits filled with 0s as shown in Figure 21-54.

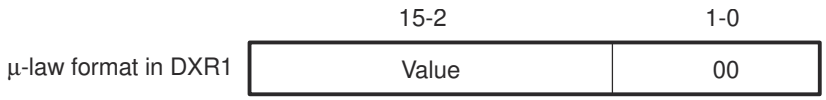



**Figure 21-54. µ-Law Transmit Data Companding Format**

For transmission using A-law compression, make sure the 13 data bits are left-justified in DXR1, with the remaining three low-order bits filled with 0s as shown in Figure 21-55.

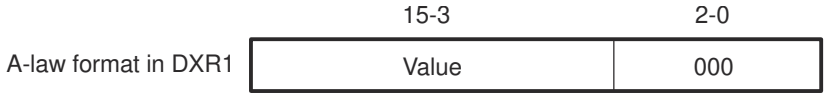



**Figure 21-55. A-Law Transmit Data Companding Format**

#### 21.9.12.3 Capability to Companding Internal Data

If the McBSP is otherwise unused (the serial port transmit and receive sections are reset), the companding hardware can compand internal data. See Section 21.3.2.2.

#### 21.9.12.4 Option to Transmit LSB First

Normally, the McBSP transmit or receives all data with the most significant bit (MSB) first. However, certain 8-bit data protocols (that do not use companded data) require the least significant bit (LSB) to be transferred first. If you set XCOMPAND = 01b in XCR2, the bit ordering of 8-bit words is reversed (LSB first) before being sent from the serial port. Similar to companding, this feature is enabled only if the appropriate word length bits are set to 0, indicating that 8-bit words are to be transferred serially. If either phase of the frame does not have an 8-bit word length, the McBSP assumes the word length is eight bits and LSB-first ordering is done.

### *21.9.13 Transmit Data Delay*

The XDATDLY bits (see Table 21-60) determine the length of the data delay for the transmit frame.

**Table 21-60. Register Bits Used to Set the Transmit Data Delay**

| Register | Bit | Name | Function | | Type | Reset Value |
|---|---|---|---|---|---|---|
| XCR2 | 1-0 | XDATDLY | Transmitter data delay | | R/W | 00 |
| | | | XDATDLY = 00 | 0-bit data delay | | |
| | | | XDATDLY = 01 | 1-bit data delay | | |
| | | | XDATDLY = 10 | 2-bit data delay | | |
| | | | XDATDLY = 11 | Reserved | | |

#### 21.9.13.1 Data Delay

The start of a frame is defined by the first clock cycle in which frame synchronization is found to be active. The beginning of actual data reception or transmission with respect to the start of the frame can be delayed if necessary. This delay is called data delay.

XDATDLY specifies the data delay for transmission. The range of programmable data delay is zero to two bit-clocks (XDATDLY = 00b-10b), as described in Table 21-60 and Figure 21-56. In this figure, the data transferred is an 8-bit value with bits labeled B7, B6, B5, and so on. Typically a 1-bit delay is selected, because data often follows a 1-cycle active frame-synchronization pulse.



Copyright © 2024 Texas Instruments Incorporated

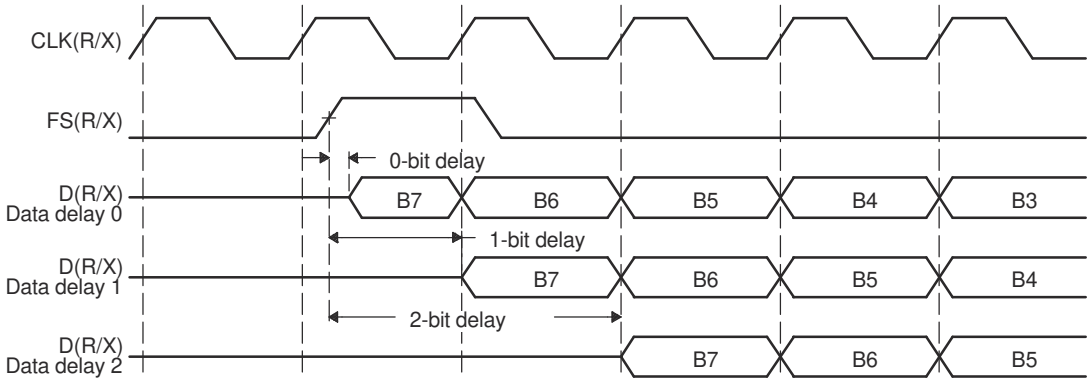



**Figure 21-56. Range of Programmable Data Delay**

#### 21.9.13.2 0-Bit Data Delay

Normally, a frame-synchronization pulse is detected or sampled with respect to an edge of serial clock internal CLK(R/X). Thus, on the following cycle or later (depending on the data delay value), data can be received or transmitted. However, in the case of 0-bit data delay, the data must be ready for reception and/or transmission on the same serial clock cycle.

For reception this problem is solved because receive data is sampled on the first falling edge of MCLKR where an active-high internal FSR is detected. However, data transmission must begin on the rising edge of the internal CLKX clock that generated the frame synchronization. Therefore, the first data bit is assumed to be present in XSR1, and thus DX. The transmitter then asynchronously detects the frame synchronization, FSX, going active high and immediately starts driving the first bit to be transmitted on the DX pin.

#### 21.9.13.3 2-Bit Data Delay

A data delay of two bit-periods allows the serial port to interface to different types of T1 framing devices where the data stream is preceded by a framing bit. During reception of such a stream with data delay of two bits (framing bit appears after a 1-bit delay and data appears after a 2-bit delay), the serial port essentially discards the framing bit from the data stream, as shown in Figure 21-57. In this figure, the data transferred is an 8-bit value with bits labeled B7, B6, B5, and so on.

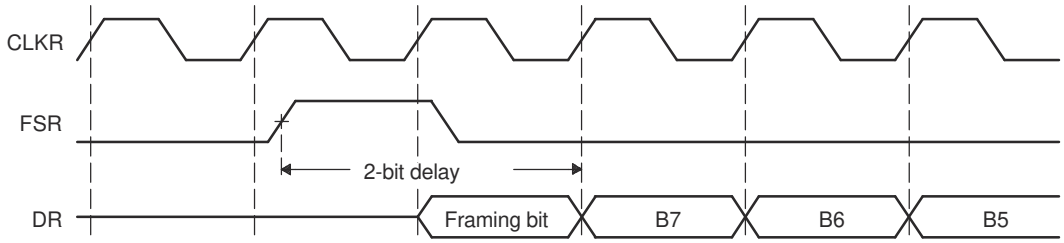



**Figure 21-57. 2-Bit Data Delay Used to Skip a Framing Bit**



Copyright © 2024 Texas Instruments Incorporated

### 21.9.14 Transmit DXENA Mode

Table 21-61 shows which register bit enables the Transmit DXENA (DX Delay Enabler) Mode.

**Table 21-61. Register Bit Used to Set the Transmit DXENA (DX Delay Enabler) Mode**

| Register | Bit | Name | Function | | Type | Reset Value |
|---|---|---|---|---|---|---|
| SPCR1 | 7 | DXENA | DX delay enabler mode | | R/W | 0 |
| | | | DXENA = 0 | DX delay enabler is off. | | |
| | | | DXENA = 1 | DX delay enabler is on. | | |

The DXENA bit controls the delay enabler on the DX pin. Set DXENA to enable an extra delay for turn-on time. This bit does not control the data itself, so only the first bit is delayed.

If you tie together the DX pins of multiple McBSPs, make sure DXENA = 1 to avoid having more than one McBSP transmit on the data line at one time.

### 21.9.15 Transmit Interrupt Mode

The transmitter interrupt (XINT) signals the CPU of changes to the serial port status. Four options exist for configuring this interrupt. The options are set by the transmit interrupt mode bits, XINTM, in SPCR2.

**Table 21-62. Register Bits Used to Set the Transmit Interrupt Mode**

| Register | Bit | Name | Function | | Type | Reset Value |
|---|---|---|---|---|---|---|
| SPCR2 | 5-4 | XINTM | Transmit interrupt mode | | R/W | 00 |
| | | | XINTM = 00 | XINT generated when XRDY changes from 0 to 1. | | |
| | | | XINTM = 01 | XINT generated by an end-of-block or end-of-frame condition in a transmit multichannel selection mode. In any of the transmit multichannel selection modes, interrupt after every 16-channel block boundary has been crossed within a frame and at the end of the frame. For details, see Section 21.6.8. In any other serial transfer case, this setting is not applicable and, therefore, no interrupts are generated. | | |
| | | | XINTM = 10 | XINT generated by a new transmit frame-synchronization pulse. Interrupt on detection of each transmit frame-synchronization pulse. This generates an interrupt even when the transmitter is in the reset state. This is done by synchronizing the incoming frame-synchronization pulse to the CPU clock and sending the pulse to the CPU using XINT. | | |
| | | | XINTM = 11 | XINT generated when XSYNCERR is set. Interrupt on frame-synchronization error. Regardless of the value of XINTM, XSYNCERR can be read to detect this condition. For more information on using XSYNCERR, see Section 21.5.6. | | |



Copyright © 2024 Texas Instruments Incorporated

### 21.9.16 Transmit Frame-Synchronization Mode

Table 21-63 shows which register bits enable the Transmit Frame-Synchronization Mode.

**Table 21-63. Register Bits Used to Set the Transmit Frame-Synchronization Mode**

| Register | Bit | Name | Function | | Type | Reset Value |
|---|---|---|---|---|---|---|
| PCR | 11 | FSXM | Transmit frame-synchronization mode | | R/W | 0 |
| | | | FSXM = 0 | Transmit frame synchronization is supplied by an external source via the FSX pin. | | |
| | | | FSXM = 1 | Transmit frame synchronization is supplied by the McBSP, as determined by the FSGM bit of SRGR2. | | |
| SRGR2 | 12 | FSGM | Sample rate generator transmit frame-synchronization mode | | R/W | 0 |
| | | | Used when FSXM = 1 in PCR. | | | |
| | | | FSGM = 0 | The McBSP generates a transmit frame-synchronization pulse when the content of DXR[1,2] is copied to XSR[1,2]. | | |
| | | | FSGM = 1 | The transmitter uses frame-synchronization pulses generated by the sample rate generator. Program the FWID bits to set the width of each pulse. Program the FPER bits to set the frame-synchronization period. | | |

Table 21-64 shows how FSXM and FSGM select the source of transmit frame-synchronization pulses. The three choices are:

- External frame-synchronization input
- Sample rate generator frame-synchronization signal (FSG)
- Internal signal that indicates a DXR-to-XSR copy has been made

Table 21-64 also shows the effect of each bit setting on the FSX pin. The polarity of the signal on the FSX pin is determined by the FSXP bit.

**Table 21-64. How FSXM and FSGM Select the Source of Transmit Frame-Synchronization Pulses**

| FSXM | FSGM | Source of Transmit Frame Synchronization | FSX Pin Status |
|---|---|---|---|
| 0 | 0 or 1 | An external frame-synchronization signal enters the McBSP through the FSX pin. The signal is then inverted by FSXP before being used as internal FSX. | Input |
| 1 | 1 | Internal FSX is driven by the sample rate generator frame-synchronization signal (FSG). | Output. FSG is inverted by FSXP before being driven out on FSX pin. |
| 1 | 0 | A DXR-to-XSR copy causes the McBSP to generate a transmit frame-synchronization pulse that is 1 cycle wide. | Output. The generated frame-synchronization pulse is inverted as determined by FSXP before being driven out on FSX pin. |

#### 21.9.16.1 Other Considerations

If the sample rate generator creates a frame-synchronization signal (FSG) that is derived from an external input clock, the GSYNC bit determines whether FSG is kept synchronized with pulses on the FSR pin. For more details, see Section 21.4.3.

In the clock stop mode (CLKSTP = 10b or 11b), the McBSP can act as a master or as a slave in the SPI protocol. If the McBSP is a master and must provide a slave-enable signal ($\overline{\text{SPISTE}}$) on the FSX pin, make sure that FSXM = 1 and FSGM = 0 so that FSX is an output and is driven active for the duration of each transmission. If the McBSP is a slave, make sure that FSXM = 0 so that the McBSP can receive the slave-enable signal on the FSX pin.



Copyright © 2024 Texas Instruments Incorporated

### 21.9.17 Transmit Frame-Synchronization Polarity

Table 21-65 shows which register bits enable the Transmit Frame-Synchronization Polarity.

**Table 21-65. Register Bit Used to Set Transmit Frame-Synchronization Polarity**

| Register | Bit | Name | Function | | Type | Reset Value |
|---|---|---|---|---|---|---|
| PCR | 3 | FSXP | Transmit frame-synchronization polarity | | R/W | 0 |
| | | | FSXP = 0 | Frame-synchronization pulse FSX is active high. | | |
| | | | FSXP = 1 | Frame-synchronization pulse FSX is active low. | | |

#### 21.9.17.1 Frame Synchronization Pulses, Clock Signals, and Their Polarities

Transmit frame-synchronization pulses can be generated internally by the sample rate generator (see Section 21.4.2) or driven by an external source. The source of frame synchronization is selected by programming the mode bit, FSXM, in PCR. FSX is also affected by the FSGM bit in SRGR2. For information about the effects of FSXM and FSGM, see Section 21.9.16). Similarly, transmit clocks can be selected to be inputs or outputs by programming the mode bit, CLKXM, in the PCR (see Section 21.9.19).

When FSR and FSX are inputs (FSXM = FSRM= 0, external frame-synchronization pulses), the McBSP detects them on the internal falling edge of clock, internal MCLKR, and internal CLKX, respectively. The receive data arriving at the DR pin is also sampled on the falling edge of internal MCLKR. These internal clock signals are either derived from external source via CLK(R/X) pins or driven by the sample rate generator clock (CLKG) internal to the McBSP.

When FSR and FSX are outputs, implying that the outputs are driven by the sample rate generator, the outputs are generated (transition to their active state) on the rising edge of internal clock, CLK(R/X). Similarly, data on the DX pin is output on the rising edge of internal CLKX.

FSRP, FSXP, CLKRP, and CLKXP in the pin control register (PCR) configure the polarities of the FSR, FSX, MCLKR, and CLKX signals, respectively. All frame-synchronization signals (internal FSR, internal FSX) that are internal to the serial port are active high. If the serial port is configured for external frame synchronization (FSR/FSX are inputs to McBSP) and FSRP = FSXP = 1, the external active-low frame-synchronization signals are inverted before being sent to the receiver (internal FSR) and transmitter (internal FSX). Similarly, if internal synchronization (FSR/FSX are output pins and GSYNC = 0) is selected and the polarity bit FS(R/X)P = 1, the internal active-high frame-synchronization signals are inverted before being sent to the FS(R/X) pin.

On the transmit side, the transmit clock polarity bit, CLKXP, sets the edge used to shift and clock out transmit data. Data is always transmitted on the rising edge of internal CLKX. If CLKXP = 1 and external clocking is selected (CLKXM = 0 and CLKX is an input), the external falling-edge triggered input clock on CLKX is inverted to a rising-edge triggered clock before being sent to the transmitter. If CLKXP = 1, and internal clocking selected (CLKXM = 1 and CLKX is an output pin), the internal (rising-edge triggered) clock, internal CLKX, is inverted before being sent out on the MCLKX pin.

Similarly, the receiver can reliably sample data that is clocked with a rising edge clock (by the transmitter). The receive clock polarity bit, CLKRP, sets the edge used to sample received data. The receive data is always sampled on the falling edge of internal MCLKR. Therefore, if CLKRP = 1 and external clocking is selected (CLKRM = 0 and MCLKR is an input pin), the external rising-edge triggered input clock on MCLKR is inverted to a falling-edge triggered clock before being sent to the receiver. If CLKRP = 1 and internal clocking is selected (CLKRM = 1), the internal falling-edge triggered clock is inverted to a rising-edge triggered clock before being sent out on the MCLKR pin.

CLKRP = CLKXP in a system where the same clock (internal or external) is used to clock the receiver and transmitter. The receiver uses the opposite edge as the transmitter to make sure of valid setup and hold of data around this edge. Figure 21-58 shows how data clocked by an external serial device using a rising edge can be sampled by the McBSP receiver on the falling edge of the same clock.



Copyright © 2024 Texas Instruments Incorporated

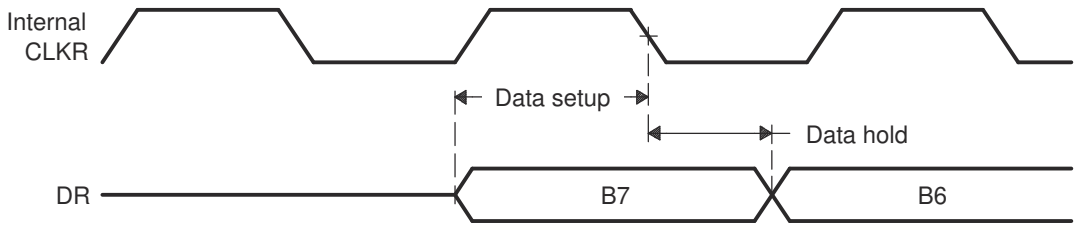



**Figure 21-58. Data Clocked Externally Using a Rising Edge and Sampled by the McBSP Receiver on a Falling Edge**

### *21.9.18 SRG Frame-Synchronization Period and Pulse Width*

Table 21-66 shows which register bits set the SRG Frame-Synchronization Period and Pulse Width.

**Table 21-66. Register Bits Used to Set SRG Frame-Synchronization Period and Pulse Width**

| Register | Bit | Name | Function | Type | Reset Value |
|---|---|---|---|---|---|
| SRGR2 | 11-0 | FPER | Sample rate generator frame-synchronization period<br>For the frame-synchronization signal FSG, (FPER + 1) determines the period from the start of a frame-synchronization pulse to the start of the next frame-synchronization pulse.<br>Range for (FPER + 1): 1 to 4096 CLKG cycles. | R/W | 0000 0000 0000 |
| SRGR1 | 15-8 | FWID | Sample rate generator frame-synchronization pulse width<br>This field plus 1 determines the width of each frame-synchronization pulse on FSG.<br>Range for (FWID + 1): 1 to 256 CLKG cycles. | R/W | 0000 0000 |

#### 21.9.18.1 Frame-Synchronization Period and Frame-Synchronization Pulse Width

The sample rate generator can produce a clock signal, CLKG, and a frame-synchronization signal, FSG. If the sample rate generator is supplying receive or transmit frame synchronization, you must program the bit fields FPER and FWID.

On FSG, the period from the start of a frame-synchronization pulse to the start of the next pulse is (FPER + 1) CLKG cycles. The 12 bits of FPER allow a frame-synchronization period of 1 to 4096 CLKG cycles, which allows up to 4096 data bits per frame. When GSYNC = 1, FPER is a don't care value.

Each pulse on FSG has a width of (FWID + 1) CLKG cycles. The eight bits of FWID allow a pulse width of 1 to 256 CLKG cycles. It is recommended that FWID be programmed to a value less than the programmed word length.

The values in FPER and FWID are loaded into separate down-counters. The 12-bit FPER counter counts down the generated clock cycles from the programmed value (4095 maximum) to 0. The 8-bit FWID counter counts down from the programmed value (255 maximum) to 0.

Figure 21-59 shows a frame-synchronization period of 16 CLKG periods (FPER = 15 or 00001111b) and a frame-synchronization pulse with an active width of 2 CLKG periods (FWID = 1).

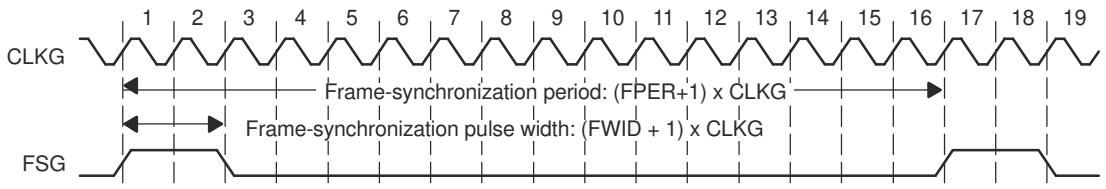



**Figure 21-59. Frame of Period 16 CLKG Periods and Active Width of 2 CLKG Periods**



Copyright © 2024 Texas Instruments Incorporated

When the sample rate generator comes out of reset, FSG is in the inactive state. Then, when GRST = 1 and FSGM = 1, a frame-synchronization pulse is generated. The frame width value (FWID + 1) is counted down on every CLKG cycle until the value reaches 0, at which time FSG goes low. At the same time, the frame period value (FPER + 1) is also counting down. When this value reaches 0, FSG goes high, indicating a new frame.

### *21.9.19 Transmit Clock Mode*

Table 21-67 shows which register bits can set the Transmit Clock Mode.

**Table 21-67. Register Bit Used to Set the Transmit Clock Mode**

| Register | Bit | Name | Function | | Type | Reset Value |
|---|---|---|---|---|---|---|
| PCR | 9 | CLKXM | Transmit clock mode | | R/W | 0 |
| | | | CLKXM = 0 | The transmitter gets the clock signal from an external source via the MCLKX pin. | | |
| | | | CLKXM = 1 | The MCLKX pin is an output pin driven by the sample rate generator of the McBSP. | | |

#### 21.9.19.1 Selecting a Source for the Transmit Clock and a Data Direction for the MCLKX pin

Table 21-68 shows how the CLKXM bit selects the transmit clock and the corresponding status of the MCLKX pin. The polarity of the signal on the MCLKX pin is determined by the CLKXP bit.

**Table 21-68. How the CLKXM Bit Selects the Transmit Clock and the Corresponding Status of the MCLKX pin**

| CLKXM in PCR | Source of Transmit Clock | MCLKX pin Status |
|---|---|---|
| 0 | Internal CLKX is driven by an external clock on the MCLKX pin. CLKX is inverted as determined by CLKXP before being used. | Input |
| 1 | Internal CLKX is driven by the sample rate generator clock, CLKG. | Output. CLKG, inverted as determined by CLKXP, is driven out on CLKX. |

#### 21.9.19.2 Other Considerations

If the sample rate generator creates a clock signal (CLKG) that is derived from an external input clock, the GSYNC bit determines whether CLKG is kept synchronized with pulses on the FSR pin. For more details, see Section 21.4.3.

In the clock stop mode (CLKSTP = 10b or 11b), the McBSP can act as a master or as a slave in the SPI protocol. If the McBSP is a master, make sure that CLKXM = 1 so that CLKX is an output to supply the master clock to any slave devices. If the McBSP is a slave, make sure that CLKXM = 0 so that CLKX is an input to accept the master clock signal.

### *21.9.20 Transmit Clock Polarity*

Table 21-69 shows which register bits set the Transmit Clock Polarity.

**Table 21-69. Register Bit Used to Set Transmit Clock Polarity**

| Register | Bit | Name | Function | | Type | Reset Value |
|---|---|---|---|---|---|---|
| PCR | 1 | CLKXP | Transmit clock polarity | | R/W | 0 |
| | | | CLKXP = 0 | Transmit data sampled on rising edge of CLKX. | | |
| | | | CLKXP = 1 | Transmit data sampled on falling edge of CLKX. | | |



Copyright © 2024 Texas Instruments Incorporated

#### 21.9.20.1 Frame Synchronization Pulses, Clock Signals, and Their Polarities

Transmit frame-synchronization pulses can be either generated internally by the sample rate generator (see Section 21.4.2) or driven by an external source. The source of frame synchronization is selected by programming the mode bit, FSXM, in PCR. FSX is also affected by the FSGM bit in SRGR2. For information about the effects of FSXM and FSGM, see Section 21.9.16). Similarly, transmit clocks can be selected to be inputs or outputs by programming the mode bit, CLKXM, in the PCR (see Section 21.9.19).

When FSR and FSX are inputs (FSXM = FSRM= 0, external frame-synchronization pulses), the McBSP detects them on the internal falling edge of clock, internal MCLKR, and internal CLKX, respectively. The receive data arriving at the DR pin is also sampled on the falling edge of internal MCLKR. These internal clock signals are either derived from external source via CLK(R/X) pins or driven by the sample rate generator clock (CLKG) internal to the McBSP.

When FSR and FSX are outputs, implying that the outputs are driven by the sample rate generator, the outputs are generated (transition to their active state) on the rising edge of internal clock, CLK(R/X). Similarly, data on the DX pin is output on the rising edge of internal CLKX.

FSRP, FSXP, CLKRP, and CLKXP in the pin control register (PCR) configure the polarities of the FSR, FSX, MCLKR, and CLKX signals, respectively. All frame-synchronization signals (internal FSR, internal FSX) that are internal to the serial port are active high. If the serial port is configured for external frame synchronization (FSR/FSX are inputs to McBSP), and FSRP = FSXP = 1, the external active-low frame-synchronization signals are inverted before being sent to the receiver (internal FSR) and transmitter (internal FSX). Similarly, if internal synchronization (FSR/FSX are output pins and GSYNC = 0) is selected, the internal active-high frame-synchronization signals are inverted, if the polarity bit FS(R/X)P = 1, before being sent to the FS(R/X) pin.

On the transmit side, the transmit clock polarity bit, CLKXP, sets the edge used to shift and clock out transmit data. Data is always transmitted on the rising edge of internal CLKX. If CLKXP = 1 and external clocking is selected (CLKXM = 0 and CLKX is an input), the external falling-edge triggered input clock on CLKX is inverted to a rising-edge triggered clock before being sent to the transmitter. If CLKXP = 1 and internal clocking is selected (CLKXM = 1 and CLKX is an output pin), the internal (rising-edge triggered) clock, internal CLKX, is inverted before being sent out on the MCLKX pin.

Similarly, the receiver can reliably sample data that is clocked with a rising edge clock (by the transmitter). The receive clock polarity bit, CLKRP, sets the edge used to sample received data. The receive data is always sampled on the falling edge of internal MCLKR. Therefore, if CLKRP = 1 and external clocking is selected (CLKRM = 0 and CLKR is an input pin), the external rising-edge triggered input clock on CLKR is inverted to a falling-edge triggered clock before being sent to the receiver. If CLKRP = 1 and internal clocking is selected (CLKRM = 1), the internal falling-edge triggered clock is inverted to a rising-edge triggered clock before being sent out on the MCLKR pin.

CLKRP = CLKXP in a system where the same clock (internal or external) is used to clock the receiver and transmitter. The receiver uses the opposite edge as the transmitter to make sure of valid setup and hold of data around this edge (see Figure 21-58).

Figure 21-60 shows how data clocked by an external serial device using a rising edge can be sampled by the McBSP receiver on the falling edge of the same clock.

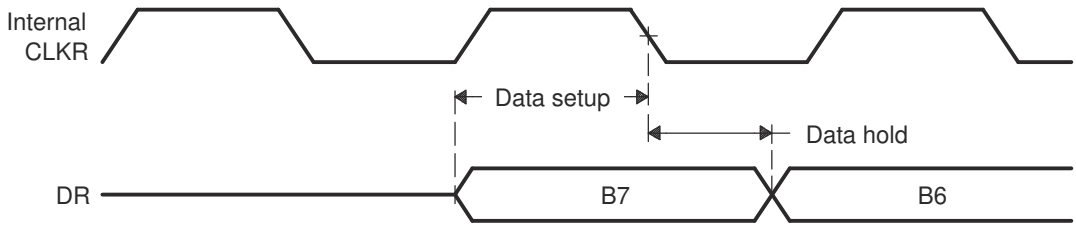



**Figure 21-60. Data Clocked Externally Using a Rising Edge and Sampled by the McBSP Receiver on a Falling Edge**



Copyright © 2024 Texas Instruments Incorporated

## 21.10 Emulation and Reset Considerations

This section covers the following topics:

- How to program McBSP response to a breakpoint in the high-level language debugger (see Section 21.10.1)
- How to reset and initialize the various parts of the McBSP (see Section 21.10.2)

### *21.10.1 McBSP Emulation Mode*

FREE and SOFT are special emulation bits in SPCR2 that determine the state of the McBSP when a breakpoint is encountered in the high-level language debugger. If FREE = 1, the clock continues to run upon a software breakpoint and data is still shifted out. When FREE = 1, the SOFT bit is a *don't care.*

If FREE = 0, the SOFT bit takes effect. If SOFT = 0 when breakpoint occurs, the clock stops immediately, aborting a transmission. If SOFT = 1 and a breakpoint occurs while transmission is in progress, the transmission continues until completion of the transfer and then the clock halts. These options are listed in Table 21-70.

The McBSP receiver functions in a similar fashion. If a mode other than the immediate stop mode (SOFT = FREE = 0) is chosen, the receiver continues running and an overrun error is possible.

**Table 21-70. McBSP Emulation Modes Selectable with FREE and SOFT Bits of SPCR2**

| FREE | SOFT | McBSP Emulation Mode |
|---|---|---|
| 0 | 0 | Immediate stop mode (reset condition).<br>The transmitter or receiver stops immediately in response to a breakpoint. |
| 0 | 1 | Soft stop mode<br>When a breakpoint occurs, the transmitter stops after completion of the current word. The receiver is not affected. |
| 1 | 0 or 1 | Free run mode<br>The transmitter and receiver continue to run when a breakpoint occurs. |

### *21.10.2 Resetting and Initializing McBSPs*

This section discusses in greater depth the MCBSP Reset and Initialization configurations.

#### 21.10.2.1 McBSP Pin States: DSP Reset Versus Receiver/Transmitter Reset

Table 21-71 shows the state of McBSP pins when the serial port is reset due to direct receiver or transmitter reset on the 2833x device.

**Table 21-71. Reset State of Each McBSP Pin**

| Pin | Possible States[1] | State Forced by Device Reset | State Forced by Receiver/Transmitter Reset |
|---|---|---|---|
| **Receiver reset (RRST = 0 and GRST = 1)** | | | |
| MDRx | I | GPIO-input | Input |
| MCLKRx | I/O/Z | GPIO-input | Known state if input; MCLKR running if output |
| MFSRx | I/O/Z | GPIO-input | Known state if input; FSRP inactive state if output |
| **Transmitter reset (XRST = 0 and GRST = 1)** | | | |
| MDXx | O/Z | GPIO Input | High impedance |
| MCLKXx | I/O/Z | GPIO-input | Known state if input; CLKX running if output |
| MFSXx | I/O/Z | GPIO-input | Known state if input; FSXP inactive state if output |

(1) In Possible States column, I = Input, O = Output, Z = High impedance. In the C28x family, at device reset, all I/Os default to GPIO function and generally as inputs.



Copyright © 2024 Texas Instruments Incorporated

#### 21.10.2.2 Device Reset, McBSP Reset, and Sample Rate Generator Reset

When the McBSP is reset in either of the above two ways, the machine is reset to its initial state, including reset of all counters and status bits. The receive status bits include RFULL, RRDY, and RSYNCERR. The transmit status bits include XEMPTY, XRDY, and XSYNCERR.

- Device reset. When the whole DSP is reset ( $\overline{\text{XRS}}$ signal is driven low), all McBSP pins are in GPIO mode. When the device is pulled out of reset, the clock to the McBSP modules remains disabled.
- McBSP reset. When the receiver and transmitter reset bits, RRST and XRST, are loaded with 0s, the respective portions of the McBSP are reset and activity in the corresponding section of the serial port stops. Input-only pins such as MDRx, and all other pins that are configured as inputs are in a known state. The MFSRx and MFSXx pins are driven to their inactive state if they are not outputs. If the MCLKR and MCLKX pins are programmed as outputs, they are driven by CLKG, provided that GRST = 1. Lastly, the MDXx pin is in the high-impedance state when the transmitter and/or the device is reset.

  During normal operation, the sample rate generator is reset if the GRST bit is cleared. GRST must be 0 only when neither the transmitter nor the receiver is using the sample rate generator. In this case, the internal sample rate generator clock (CLKG) and its frame-synchronization signal (FSG) are driven inactive low.

  When the sample rate generator is not in the reset state (GRST = 1), pins MFSRx and MFSXx are in an inactive state when RRST = 0 and XRST = 0, respectively, even if they are outputs driven by FSG. This ensures that when only one portion of the McBSP is in reset, the other portion can continue operation when GRST = 1 and its frame synchronization is driven by FSG.
- Sample rate generator reset. The sample rate generator is reset when GRST is loaded with 0.

  When neither the transmitter nor the receiver is fed by CLKG and FSG, you can reset the sample rate generator by clearing GRST. In this case, CLKG and FSG are driven inactive low. If you then set GRST, CLKG starts and runs as programmed. Later, if GRST = 1, FSG pulses active high after the programmed number of CLKG cycles has elapsed.



Copyright © 2024 Texas Instruments Incorporated

#### 21.10.2.3 McBSP Initialization Procedure

The serial port initialization procedure is as follows:

1. Make XRST = RRST = GRST = 0 in SPCR[1,2]. If coming out of a device reset, this step is not required.
2. While the serial port is in the reset state, program only the McBSP configuration registers (not the data registers) as required.
3. Wait for two clock cycles. This makes sure of proper internal synchronization.
4. Set up data acquisition as required (such as writing to DXR[1,2]).
5. Make XRST = RRST = 1 to enable the serial port. Make sure that as you set these reset bits, you do not modify any of the other bits in SPCR1 and SPCR2. Otherwise, you change the configuration you selected in step 2.
6. Set FRST = 1, if internally generated frame synchronization is required.
7. Wait two clock cycles for the receiver and transmitter to become active.

Alternatively, on either write (step 1 or 5), the transmitter and receiver can be placed in or taken out of reset individually by modifying the desired bit.

The previous procedure for reset/initialization can be applied in general when the receiver or transmitter must be reset during normal operation and when the sample rate generator is not used for either operation.

**Note**

1. The necessary duration of the active-low period of XRST or RRST is at least two MCLKR/CLKX cycles.
2. The appropriate bits in serial port configuration registers SPCR[1,2], PCR, RCR[1,2], XCR[1,2], and SRGR[1,2] must only be modified when the affected portion of the serial port is in the reset state.
3. In most cases, the data transmit registers (DXR[1,2]) must be loaded by the CPU or by the DMA controller only when the transmitter is enabled (XRST = 1). An exception to this rule is when these registers are used for companding internal data (see Section 21.3.2.2).
4. The bits of the channel control registers—MCR[1,2], RCER[A-H], XCER[A-H]—can be modified at any time as long as the bits are not being used by the current reception/transmission in a multichannel selection mode.



Copyright © 2024 Texas Instruments Incorporated

#### 21.10.2.4 Resetting the Transmitter While the Receiver is Running

Example 21-1 shows values in the control registers that reset and configure the transmitter while the receiver is running.

***Example 21-1. Resetting and Configuring McBSP Transmitter While McBSP Receiver Running***

```
SPCR1 = 0001h SPCR2 = 0030h ;
The receiver is running with the receive interrupt (RINT) triggered by the receiver ready bit
(RRDY).;
The transmitter is in its reset state.;
The transmit interrupt (XINT) will be triggered by the transmit frame-sync error bit (XSYNCERR).;
PCR = 0900h ;
Transmit frame synchronization is generated internally according to the FSGM bit of SRGR2.;
The transmit clock is driven by an external source. ;
The receive clock continues to be driven by sample rate generator. The input clock ;
of the sample rate generator is supplied by the CPU clock. ;
SRGR1 = 0001h SRGR2 = 2000h ;
The CPU clock is the input clock for the sample rate generator. The sample ;
rate generator divides the CPU clock by 2 to generate its output clock (CLKG). ;
Transmit frame synchronization is tied to the automatic copying of data from ;
the DXR(s) to the XSR(s). ;
XCR1 = 0740h XCR2 = 8321h ;
The transmit frame has two phases. Phase 1 has eight 16-bit words. ;
Phase 2 has four 12-bit words. There is 1-bit data delay between the start of a ;
frame-sync pulse and the first data bit transmitted. ;
SPCR2 = 0031h ;
The transmitter is taken out of reset.
```

## 21.11 Data Packing Examples

This section shows two ways to implement data packing in the McBSP.

### *21.11.1 Data Packing Using Frame Length and Word Length*

Frame length and word length can be manipulated to effectively pack data. For example, consider a situation where four 8-bit words are transferred in a single-phase frame as shown in Figure 21-61. In this case:

- (R/X)PHASE = 0: Single-phase frame
- (R/X)FRLEN1 = 0000011b: 4-word frame
- (R/X)WDLEN1 = 000b: 8-bit words

Four 8-bit data words are transferred to and from the McBSP by the CPU or by the DMA controller. Thus, four reads from DRR1 and four writes to DXR1 are necessary for each frame.

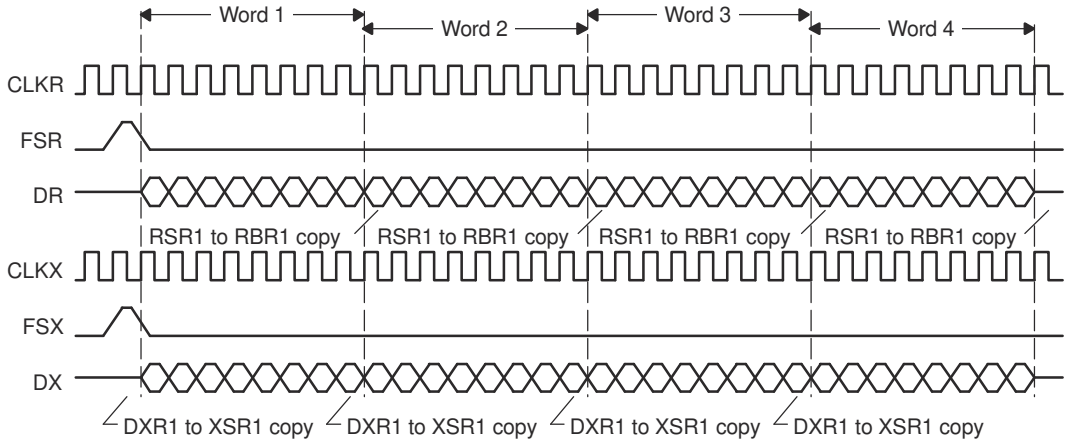



**Figure 21-61. Four 8-Bit Data Words Transferred To/From the McBSP**



Copyright © 2024 Texas Instruments Incorporated

This data can also be treated as a single-phase frame consisting of one 32-bit data word, as shown in Figure 21-62. In this case:

- (R/X)PHASE = 0: Single-phase frame
- (R/X)FRLEN1 = 0000000b: 1-word frame
- (R/X)WDLEN1 = 101b: 32-bit word

Two 16-bit data words are transferred to and from the McBSP by the CPU or DMA controller. Thus, two reads, from DRR2 and DRR1, and two writes, to DXR2 and DXR1, are necessary for each frame. This results in only half the number of transfers compared to the previous case. This manipulation reduces the percentage of bus time required for serial port data movement.

**Note**

When the word length is larger than 16 bits, make sure you access DRR2/DXR2 before you access DRR1/DXR1. McBSP activity is tied to accesses of DRR1/DXR1. During the reception of 24-bit or 32-bit words, read DRR2 and then read DRR1. Otherwise, the next RBR[1,2]-to-DRR[1,2] copy occurs before DRR2 is read. Similarly, during the transmission of 24-bit or 32-bit words, write to DXR2 and then write to DXR1. Otherwise, the next DXR[1,2]-to-XSR[1,2] copy occurs before DXR2 is loaded with new data.

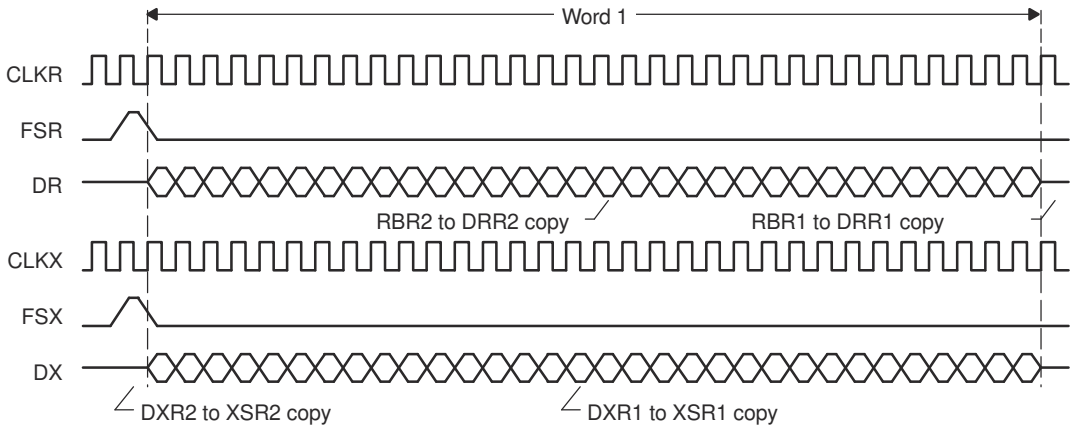



**Figure 21-62. One 32-Bit Data Word Transferred To/From the McBSP**



Copyright © 2024 Texas Instruments Incorporated

### 21.11.2 Data Packing Using Word Length and the Frame-Synchronization Ignore Function

When there are multiple words per frame, you can implement data packing by increasing the word length (defining a serial word with more bits) and by ignoring frame-synchronization pulses. First, consider Figure 21-63, which shows the McBSP operating at the maximum packet frequency. Here, each frame only has a single 8-bit word. Notice the frame-synchronization pulse that initiates each frame transfer for reception and for transmission. For reception, this configuration requires one read operation for each word. For transmission, this configuration requires one write operation for each word.

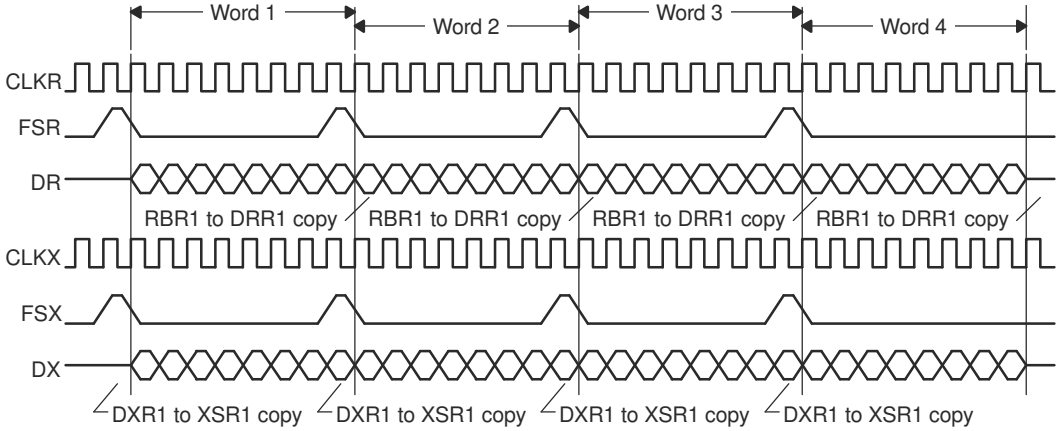



**Figure 21-63. 8-Bit Data Words Transferred at Maximum Packet Frequency**

Figure 21-64 shows the McBSP configured to treat this data stream as a continuous 32-bit word. In this example, the McBSP responds to an initial frame-synchronization pulse. However, (R/X)FIG = 1 so that the McBSP ignores subsequent pulses. Only two read transfers or two write transfers are needed every 32 bits. This configuration effectively reduces the required bus bandwidth to half the bandwidth needed to transfer four 8-bit words.

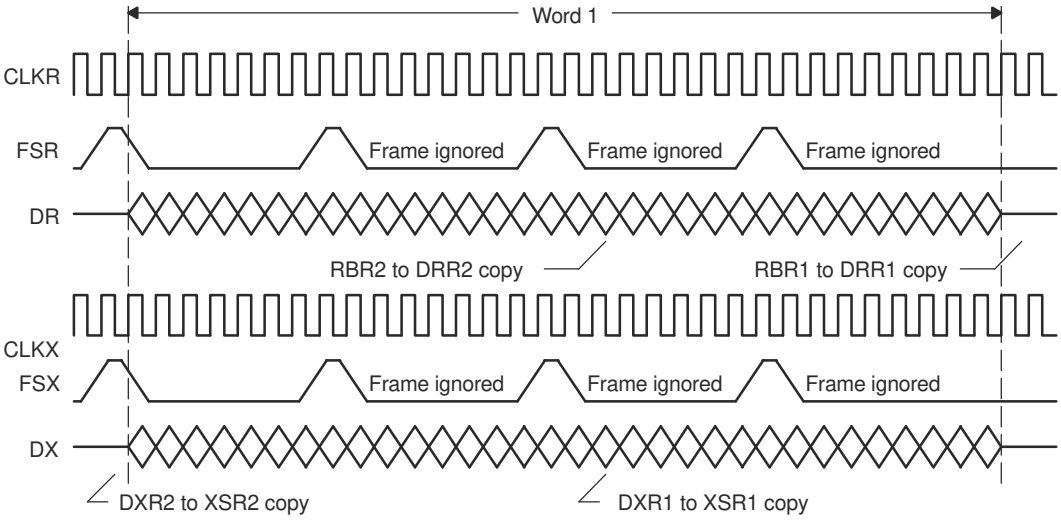



**Figure 21-64. Configuring the Data Stream of Figure 21-63 as a Continuous 32-Bit Word**



Copyright © 2024 Texas Instruments Incorporated

## 21.12 Interrupt Generation

McBSP registers can be programmed to receive and transmit data through DRR2/DRR1 and DXR2/DXR1 registers, respectively. The CPU can directly access these registers to move data from memory to these registers. Interrupt signals are based on these register pair contents and the related flags. MRINT/MXINT generates CPU interrupts for receive and transmit conditions.

### *21.12.1 McBSP Receive Interrupt Generation*

In the McBSP module, data receive and error conditions generate two sets of interrupt signals. One set is used for the CPU and the other set is for DMA.

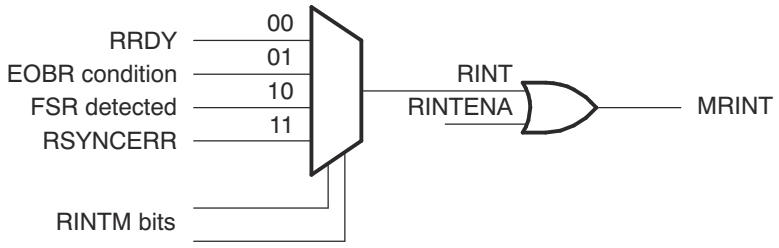



**Figure 21-65. Receive Interrupt Generation**

**Table 21-72. Receive Interrupt Sources and Signals**

| McBSP Interrupt Signal | Interrupt Flags | Interrupt Enables in SPCR1 (RINTM Bits) | Interrupt Enables | Type of Interrupt | Interrupt Line |
|---|---|---|---|---|---|
| RINT | RRDY | 00 | RINTENA | Every word receive | MRINT |
| | EOBR | 01 | RINTENA | Every 16-channel block boundary | |
| | FSR | 10 | RINTENA | On every FSR | |
| | RSYNCERR | 11 | RINTENA | Frame sync error | |

**Note**

Since X/RINT, X/REVTA, and X/RXFFINT share the same CPU interrupt, it is recommended that all applications use one of the above selections for interrupt generation. If multiple interrupt enables are selected at the same time, there is a likelihood of interrupts being masked or not recognized.



Copyright © 2024 Texas Instruments Incorporated

### 21.12.2 McBSP Transmit Interrupt Generation

McBSP module data transmit and error conditions generate two sets of interrupt signals. One set is used for the CPU and the other set is for DMA.

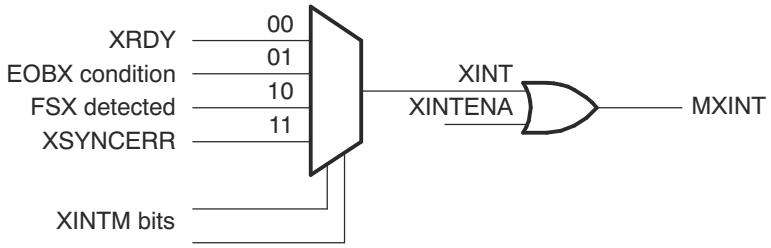



**Figure 21-66. Transmit Interrupt Generation**

**Table 21-73. Transmit Interrupt Sources and Signals**

| McBSP Interrupt Signal | Interrupt Flags | Interrupt Enables in SPCR2 (XINTM Bits) | Interrupt Enables | Type of Interrupt | Interrupt Line |
|---|---|---|---|---|---|
| XINT | XRDY | 00 | XINTENA | Every word transmit | MXINT |
| | EOBX | 01 | XINTENA | Every 16-channel block boundary | |
| | FSX | 10 | XINTENA | On every FSX | |
| | XSYNCERR | 11 | XINTENA | Frame sync error | |

### 21.12.3 Error Flags

The McBSP has several error flags both on receive and transmit channel. Table 21-74 explains the error flags and their meaning.

**Table 21-74. Error Flags**

| Error Flags | Function |
|---|---|
| RFULL | Indicates DRR2/DRR1 are not read and RXR register is overwritten |
| RSYNCERR | Indicates unexpected frame-sync condition, current data reception will abort and restart. Use RINTM bit 11 for interrupt generation on this condition. |
| XSYNCERR | Indicates unexpected frame-sync condition, current data transmission will abort and restart. Use XINTM bit 11 for interrupt generation on this condition. |



Copyright © 2024 Texas Instruments Incorporated

## 21.13 McBSP Modes

McBSP, in its normal mode, communicates with various types of Codecs with variable word size. Apart from this mode, the McBSP uses time-division multiplexed (TDM) data stream while communicating with other McBSPs or serial devices. The multichannel mode provides flexibility while transmitting/receiving selected channels or all the channels in a TDM stream.

Table 21-75 provides a quick reference to McBSP mode selection.

**Table 21-75. McBSP Mode Selection**

| No. | McBSP Word Size | Register Bits Used for Mode Selection | | | | Mode and Function Description |
|---|---|---|---|---|---|---|
| | | MCR1 bit 9,0 | | MCR2 bit 9,1,0 | | |
| | | RMCME | RMCM | XMCME | XMCM | |
| | | | | | | **Normal Mode** |
| 1 | 8/12/16/20/24/32 bit words | 0 | 0 | 0 | 0 | All types of Codec interface will use this selection |
| | | | | | | **Multichannel Mode** |
| 2 | 8-bit words | | | | | 2 Partition or 32-channel Mode |
| | | 0 | 1 | 0 | 1 | All channels are disabled, unless selected in X/RCERA/B |
| | | 0 | 1 | 0 | 10 | All channels are enabled,but masked unless selected in X/RCERA/B |
| | | 0 | 1 | 0 | 11 | Symmetric transmit, receive |
| | | | | | | 8 Partition or 128 Channel Mode Transmit/Receive |
| | | | | | | Channels selected by X/RCERA to X/RCERH bits |
| | | | | | | **Multichannel Mode is ON** |
| | | 1 | 1 | 1 | 1 | All channels are disabled, unless selected in XCERs |
| | | 1 | 1 | 1 | 10 | All channels are enabled, but masked unless selected in XCERs |
| | | 1 | 1 | 1 | 11 | Symmetric transmit, receive |
| | | | | | | Continuous mode, transmit |
| | | 1 | 0 | 1 | 0 | **Multi-Channel Mode is OFF** |
| | | | | | | All 128 channels are active and enabled |

## 21.14 Special Case: External Device is the Transmit Frame Master

Care must be taken if the transmitter expects a frame sync from an external device. After the transmitter comes out of reset (XRST = 1), the transmitter waits for a frame sync from the external device. If the first frame sync arrives very shortly after the transmitter is enabled, the CPU or DMA controller cannot have a chance to service DXR. In this case, the transmitter shifts out the default data in XSR instead of the desired value, which has not yet arrived in DXR. This causes problems in some applications, as the first data element in the frame is invalid. The data stream appears element-shifted (the first data word can appear in the second channel instead of the first).

To make sure of proper operation when the external device is the frame master, you must verify that DXR is already serviced with the first word when a frame sync occurs. To do so, you can keep the transmitter in reset until the first frame sync is detected. Upon detection of the first frame sync, the McBSP generates an interrupt to the CPU. Within the interrupt service routine, the transmitter is taken out of reset (XRST = 1). This verifies that the transmitter does not begin data transfers at the data pin during the first frame sync period. This also provides almost an entire frame period for the DSP to service DXR with the first word before the second frame sync occurs. The transmitter only begins data transfers upon receiving the second frame sync. At this point, DXR is already serviced with the first word.



Copyright © 2024 Texas Instruments Incorporated

The interrupt service routine must first be setup, then follow this modified procedure for proper initialization:

1. Make sure that no portion of the McBSP is using the internal sample rate generator signal CLKG and the internal frame sync generator signal FSG (GRST = FRST = 0). The respective portion of the McBSP needs to be in reset (XRST = 0 and/or RRST = 0).
2. Program SRGR and other control registers as required. Make sure the internal sample rate generator and the internal frame sync generator are still in reset (GRST = FRST = 0). Also make sure the respective portion of the McBSP is still in reset in this step (XRST = 0 and/or RRST = 0).
3. Program the XINTM bits to 2h in SPCR to generate an interrupt to the CPU upon detection of a transmit frame sync. Do not enable the XINT interrupt in the interrupt enable register (IER) in this step.
4. Wait for proper internal synchronization. If the external device provides the bit clock, wait for two CLKR or CLKX cycles. If the McBSP generates the bit clock as a master clock, wait for two CLKSRG cycles. In this case, the clock source to the sample rate generator (CLKSRG) is selected by the CLKSM bit in SRGR.
5. Skip this step if the bit clock is provided by the external device. This step only applies if the McBSP is the bit master clock and the internal sample rate generator is used.
   a. Start the sample rate generator by setting the GRST bit to 1. Wait two CLKG bit clocks for synchronization. CLKG is the output of the sample rate generator.
   b. On the next rising edge of CLKSRG, CLKG transitions to 1 and starts clocking with a frequency equal to 1/(CLKGDV + 1) of the sample rate generator source clock CLKSRG.
6. A transmit sync error (XSYNCERR) can occur when enabled for the first time after device reset. The purpose of this step is to clear any potential XSYNCERR that occurs on the transmitter at this time:
   a. Set the XRST bit to 1 to enable the transmitter.
   b. Wait for any unexpected frame sync error to occur. If the external device provides the bit clock, wait for two CLKR or CLKX cycles. If the McBSP generates the bit clock as a master clock, wait for two CLKG cycles. The unexpected frame sync error (XSYNCERR), if any, occurs within this time period.
   c. Disable the transmitter (XRST = 0). This clears any outstanding XSYNCERR.
7. Setup data acquisition as required:
   a. If the DMA controller is used to service the McBSP, setup data acquisition as desired and start the DMA controller in this step, before the McBSP is taken out of reset.
   b. If CPU interrupt is used to service the McBSP, no action is required in this step.
   c. If CPU polling is used to service the McBSP, no action is required in this step.
8. Enable the XINT interrupt by setting the corresponding bit in the interrupt enable register (IER). In this step, the McBSP transmitter is still in reset. Upon detection of the first transmit frame sync from the external device, the McBSP generates an interrupt to the CPU and the DSP enters the interrupt service routine (ISR). The ISR needs to perform these tasks in this order:
   a. Modify the XINTM bits to the value desired for normal McBSP operations. If CPU interrupt is used to service the McBSP in normal operations, make sure that the XINTM bits are modified to 0 to detect the McBSP XRDY event. If no McBSP interrupt is desired in normal operations, disable future McBSP-to-CPU interrupt in the interrupt enable register (IER).
   b. Set the XRST bit and/or the RRST bit to 1 to enable the respective portion of the McBSP. The McBSP is now ready to transmit and/or receive.
9. Service the McBSP:
   a. If CPU polling is used to service the McBSP in normal operations, CPU polling can do so upon exit from the ISR.
   b. If CPU interrupt is used to service the McBSP in normal operations, upon XRDY interrupt service routine is entered. The ISR must be setup to verify that XRDY = 1 and service the McBSP accordingly.
   c. If DMA controller is used to service the McBSP in normal operations, the DMA controller services the McBSP automatically upon receiving the XEVT and/or REVT.
10. Upon detection of the second frame sync, DXR is already serviced and the transmitter is ready to transmit the valid data. The receiver is also serviced properly by the DSP.



Copyright © 2024 Texas Instruments Incorporated

## 21.15 Software

### *21.15.1 MCBSP Examples*

NOTE: These examples are located in the C2000Ware installation at the following location:
C2000Ware_VERSION#/driverlib/DEVICE_GPN/examples/CORE_IF_MULTICORE/mcbsp

Cloud access to these examples is available at the following link: dev.ti.com C2000Ware Examples.

## 21.16 McBSP Registers

This section describes the McBSP registers.

### *21.16.1 McBSP Base Addresses*

**Table 21-76. McBSP Base Address Table**

| Device Registers | Register Name | Start Address | End Address |
|---|---|---|---|
| McbspaRegs | MCBSP_REGS | 0x0000_6000 | 0x0000_603F |
| McbspbRegs | MCBSP_REGS | 0x0000_6040 | 0x0000_607F |



Copyright © 2024 Texas Instruments Incorporated

### 21.16.2 McBSP_REGS Registers

Table 21-77 lists the memory-mapped registers for the McBSP_REGS registers. All register offset addresses not listed in Table 21-77 should be considered as reserved locations and the register contents should not be modified.

**Table 21-77. MCBSP_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | DRR2 | Data receive register bits 31-16 | | Go |
| 1h | DRR1 | Data receive register bits 15-0 | | Go |
| 2h | DXR2 | Data transmit register bits 31-16 | | Go |
| 3h | DXR1 | Data transmit register bits 15-0 | | Go |
| 4h | SPCR2 | Serial port control register 2 | | Go |
| 5h | SPCR1 | Serial port control register 1 | | Go |
| 6h | RCR2 | Receive Control register 2 | | Go |
| 7h | RCR1 | Receive Control register 1 | | Go |
| 8h | XCR2 | Transmit Control register 2 | | Go |
| 9h | XCR1 | Transmit Control register 1 | | Go |
| Ah | SRGR2 | Sample rate generator register 2 | | Go |
| Bh | SRGR1 | Sample rate generator register 1 | | Go |
| Ch | MCR2 | Multi-channel control register 2 | | Go |
| Dh | MCR1 | Multi-channel control register 1 | | Go |
| Eh | RCERA | Receive channel enable partition A | | Go |
| Fh | RCERB | Receive channel enable partition B | | Go |
| 10h | XCERA | Transmit channel enable partition A | | Go |
| 11h | XCERB | Transmit channel enable partition B | | Go |
| 12h | PCR | Pin Control register | | Go |
| 13h | RCERC | Receive channel enable partition C | | Go |
| 14h | RCERD | Receive channel enable partition D | | Go |
| 15h | XCERC | Transmit channel enable partition C | | Go |
| 16h | XCERD | Transmit channel enable partition D | | Go |
| 17h | RCERE | Receive channel enable partition E | | Go |
| 18h | RCERF | Receive channel enable partition F | | Go |
| 19h | XCERE | Transmit channel enable partition E | | Go |
| 1Ah | XCERF | Transmit channel enable partition F | | Go |
| 1Bh | RCERG | Receive channel enable partition G | | Go |
| 1Ch | RCERH | Receive channel enable partition H | | Go |
| 1Dh | XCERG | Transmit channel enable partition G | | Go |
| 1Eh | XCERH | Transmit channel enable partition H | | Go |
| 23h | MFFINT | Interrupt enable | | Go |

Complex bit access types are encoded to fit into small table cells. Table 21-78 shows the codes that are used for access types in this section.

**Table 21-78. McBSP_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| Write Type | | |



Copyright © 2024 Texas Instruments Incorporated

**Table 21-78. McBSP_REGS Access Type Codes (continued)**

| Access Type | Code | Description |
|---|---|---|
| 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. |



Copyright © 2024 Texas Instruments Incorporated

### 21.16.2.1 DRR2 Register (Offset = 0h) [Reset = 0000h]

DRR2 is shown in Figure 21-67 and described in Table 21-79.

Return to the Summary Table.

DRR2 contains the upper 16 bits of the received data to be read by the CPU or DMA. DRR2 is only used if the word length is greater than 16 bits.

**Figure 21-67. DRR2 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| HWHB | | | | | | | |
| R/W-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HWLB | | | | | | | |
| R/W-0h | | | | | | | |

**Table 21-79. DRR2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | HWHB | R/W | 0h | High word high byte<br>Reset type: SYSRSn |
| 7-0 | HWLB | R/W | 0h | High word low byte<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 21.16.2.2 DRR1 Register (Offset = 1h) [Reset = 0000h]

DRR1 is shown in Figure 21-68 and described in Table 21-80.

Return to the Summary Table.

DRR1 contains the lower 16 bits of the received data to be read by either the CPU or DMA.

**Figure 21-68. DRR1 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| LWHB | | | | | | | |
| R/W-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| LWLB | | | | | | | |
| R/W-0h | | | | | | | |

**Table 21-80. DRR1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | LWHB | R/W | 0h | Low word high byte<br>Reset type: SYSRSn |
| 7-0 | LWLB | R/W | 0h | Low word low byte<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 21.16.2.3 DXR2 Register (Offset = 2h) [Reset = 0000h]

DXR2 is shown in Figure 21-69 and described in Table 21-81.

Return to the Summary Table.

DXR2 contains the upper 16 bits of the data to be transmitted after being written by the CPU or DMA. DXR2 is only used if the word length is greater than 16 bits.

**Figure 21-69. DXR2 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| HWHB | | | | | | | |
| R/W-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HWLB | | | | | | | |
| R/W-0h | | | | | | | |

**Table 21-81. DXR2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | HWHB | R/W | 0h | Low word high byte<br>Reset type: SYSRSn |
| 7-0 | HWLB | R/W | 0h | Low word low byte<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 21.16.2.4 DXR1 Register (Offset = 3h) [Reset = 0000h]

DXR1 is shown in Figure 21-70 and described in Table 21-82.

Return to the Summary Table.

DXR1 contains the lower 16 bits of the data to be transmitted after being written by the CPU or DMA.

**Figure 21-70. DXR1 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| LWHB | | | | | | | |
| R/W-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| LWLB | | | | | | | |
| R/W-0h | | | | | | | |

**Table 21-82. DXR1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | LWHB | R/W | 0h | Low word high byte<br>Reset type: SYSRSn |
| 7-0 | LWLB | R/W | 0h | Low word low byte<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 21.16.2.5 SPCR2 Register (Offset = 4h) [Reset = 0000h]

SPCR2 is shown in Figure 21-71 and described in Table 21-83.

Return to the Summary Table.

SPCR2 conatins control and status bits for various McBSP functions such as emulation modes, transmit interrupt mode control, transmitter status bits, and transmitter and other internal reset controls.

**Figure 21-71. SPCR2 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | FREE | SOFT |
| R-0h | | | | | | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| FRST | GRST | XINTM | | XSYNCERR | XEMPTY | XRDY | XRST |
| R/W-0h | R/W-0h | R/W-0h | | R/W-0h | R-0h | R-0h | R/W-0h |

**Table 21-83. SPCR2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-10 | RESERVED | R | 0h | Reserved |
| 9 | FREE | R/W | 0h | Free run bit.<br>When a breakpoint is encountered in the high-level language debugger, FREE determines whether the McBSP transmit and receive clocks continue to run or whether they are affected as determined by the SOFT bit. When one of the clocks stops, the corresponding data transfer (transmission or reception) stops.<br>Reset type: SYSRSn |
| 8 | SOFT | R/W | 0h | Soft stop bit.<br>When FREE = 0, SOFT determines the response of the McBSP transmit and receive clocks when a breakpoint is encountered in the high-level language debugger. When one of the clocks stops, the corresponding data transfer (transmission or reception) stops.<br>Reset type: SYSRSn |
| 7 | FRST | R/W | 0h | Frame-synchronization logic reset bit.<br>The sample rate generator of the McBSP includes framesynchronization logic to generate an internal frame-synchronization signal. You can use FRST to take the frame-synchronization logic into and out of its reset state. This bit has a negative polarity<br>FRST = 0 indicates the reset state.<br>Reset type: SYSRSn<br>0h (R/W) = If you read a 0, the frame-synchronization logic is in its reset state.<br>If you write a 0, you reset the frame-synchronization logic.<br>In the reset state, the frame-synchronization logic does not generate a frame-synchronization signal<br>(FSG).<br>1h (R/W) = If you read a 1, the frame-synchronization logic is enabled.<br>If you write a 1, you enable the frame-synchronization logic by taking it out of its reset state.<br>When the frame-synchronization logic is enabled (FRST = 1) and the sample rate generator as a whole is enabled (GRST = 1), the frame-synchronization logic generates the frame-synchronization signal FSG as programmed. |



Copyright © 2024 Texas Instruments Incorporated

**Table 21-83. SPCR2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 6 | GRST | R/W | 0h | Sample rate generator reset bit.<br>You can use GRST to take the McBSP sample rate generator into and out of its reset state. This bit has a negative polarity<br>GRST = 0 indicates the reset state.<br>Reset type: SYSRSn<br>0h (R/W) = If you read a 0, the sample rate generator is in its reset state.<br>If you write a 0, you reset the sample rate generator.<br>If GRST = 0 due to a reset, CLKG is driven by the CPU clock divided by 2, and FSG is driven low (inactive). If GRST = 0 due to program code, CLKG and FSG are both driven low (inactive).<br>1h (R/W) = If you read a 1, the sample rate generator is enabled.<br>If you write a 1, you enable the sample rate generator by taking it out of its reset state.<br>When enabled, the sample rate generator generates the clock signal CLKG as programmed in the sample rate generator registers. If FRST = 1, the generator also generates the frame-synchronization signal FSG as programmed in the sample rate generator registers. |
| 5-4 | XINTM | R/W | 0h | Transmit interrupt mode bits.<br>XINTM determines which event in the McBSP transmitter generates a transmit interrupt (XINT) request. If XINT is properly enabled, the CPU services the interrupt request<br>otherwise, the CPU ignores the request.<br>Reset type: SYSRSn<br>0h (R/W) = The McBSP sends a transmit interrupt (XINT) request to the CPU when the XRDY bit changes from 0 to 1, indicating that transmitter is ready to accept new data (the content of DXR[1,2] has been copied to XSR[1,2]).<br>Regardless of the value of XINTM, you can check XRDY to determine whether a word transfer is complete.<br>The McBSP sends an XINT request to the CPU when 16 enabled bits have been transmitted on the DX pin.<br>1h (R/W) = In the multichannel selection mode, the McBSP sends an XINT request to the CPU after every 16- channel block is transmitted in a frame.<br>Outside of the multichannel selection mode, no interrupt request is sent.<br>2h (R/W) = The McBSP sends an XINT request to the CPU when each transmit frame-synchronization pulse is detected. The interrupt request is sent even if the transmitter is in its reset state.<br>3h (R/W) = The McBSP sends an XINT request to the CPU when the XSYNCERR bit is set, indicating a transmit frame-synchronization error.<br>Regardless of the value of XINTM, you can check XSYNCERR to determine whether a transmit framesynchronization error occurred. |
| 3 | XSYNCERR | R/W | 0h | Transmit frame-synchronization error bit.<br>XSYNCERR is set when a transmit frame-synchronization error is detected by the McBSP. If XINTM = 11b, the McBSP sends a transmit interrupt (XINT) request to the CPU when XSYNCERR is set. The flag remains set until you write a 0 to it or reset the transmitter.<br>Reset type: SYSRSn<br>0h (R/W) = No error<br>1h (R/W) = Transmit frame-synchronization error |



Copyright © 2024 Texas Instruments Incorporated

**Table 21-83. SPCR2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 2 | XEMPTY | R | 0h | Transmitter empty bit.<br>XEMPTY is cleared when the transmitter is ready to send new data but no new data is available (transmitter-empty condition). This bit has a negative polarity<br>a transmitter-empty condition is indicated by XEMPTY = 0.<br>Reset type: SYSRSn<br>0h (R/W) = Transmitter-empty condition<br>Typically this indicates that all the bits of the current word have been transmitted but there is no new<br>data in DXR1. XEMPTY is also cleared if the transmitter is reset and then restarted.<br>1h (R/W) = No transmitter-empty condition |
| 1 | XRDY | R | 0h | Transmitter ready bit.<br>XRDY is set when the transmitter is ready to accept new data in DXR[1,2]. Specifically, XRDY is set in response to a copy from DXR1 to XSR1.<br>If the transmit interrupt mode is XINTM = 00b, the McBSP sends a transmit interrupt (XINT) request to the CPU when XRDY changes from 0 to 1.<br>Also, when XRDY changes from 0 to 1, the McBSP sends a transmit synchronization event (XEVT) signal to the DMA controller.<br>Reset type: SYSRSn<br>0h (R/W) = Transmitter not ready<br>When DXR1 is loaded, XRDY is automatically cleared.<br>1h (R/W) = Transmitter ready: DXR[1,2] is ready to accept new data.<br>If both DXRs are needed (word length larger than 16 bits), the CPU or the DMA controller must load DXR2 first and then load DXR1. As soon as DXR1 is loaded, the contents of both DXRs are copied to the transmit shift registers (XSRs), as described in the next step. If DXR2 is not loaded first, the previous content of DXR2 is passed to the XSR2 |
| 0 | XRST | R/W | 0h | Transmitter reset bit. You can use XRST to take the McBSP transmitter into and out of its reset state. This bit has a negative polarity<br>XRST = 0 indicates the reset state.<br>To read about the effects of a transmitter reset, see Section 15.10.2, Resetting and Initializing a McBSP.<br>Reset type: SYSRSn<br>0h (R/W) = If you read a 0, the transmitter is in its reset state.<br>If you write a 0, you reset the transmitter.<br>1h (R/W) = If you read a 1, the transmitter is enabled.<br>If you write a 1, you enable the transmitter by taking it out of its reset state. |



Copyright © 2024 Texas Instruments Incorporated

### 21.16.2.6 SPCR1 Register (Offset = 5h) [Reset = 0000h]

SPCR1 is shown in Figure 21-72 and described in Table 21-84.

Return to the Summary Table.

SPCR1 contains control and status bits for various McBSP functions such as digital loopback, receive data justification, clock stop mode, receive interrupt mode, DX pin delay enabler, receiver status bits, and receiver reset control.

**Figure 21-72. SPCR1 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| DLB | RJUST | | CLKSTP | | RESERVED | | |
| R/W-0h | R/W-0h | | R/W-0h | | R-0h | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DXENA | RESERVED | RINTM | | RSYNCERR | RFULL | RRDY | RRST |
| R/W-0h | R/W-0h | R/W-0h | | R/W-0h | R-0h | R-0h | R/W-0h |

**Table 21-84. SPCR1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | DLB | R/W | 0h | Digital loopback mode bit.<br>DLB disables or enables the digital loopback mode of the McBSP:<br>Reset type: SYSRSn<br>0h (R/W) = Disabled<br>Internal DR is supplied by the MDRx pin. Internal FSR and internal MCLKR can be supplied by their respective pins or by the sample rate generator, depending on the mode bits FSRM and CLKRM. Internal DX is supplied by the MDXx pin. Internal FSX and internal CLKX are supplied by their respective pins or are generated internally, depending on the mode bits FSXM and CLKXM.<br>1h (R/W) = Enabled<br>Internal receive signals are supplied by internal transmit signals:<br>MDRx connected to MDXx<br>MFSRx connected to MFSXx<br>MCLKR connected to MCLKXx<br>This mode allows you to test serial port code with a single DSP. The McBSP transmitter directly supplies data, frame synchronization, and clocking to the McBSP receiver. |
| 14-13 | RJUST | R/W | 0h | Receive sign-extension and justification mode bits.<br>During reception, RJUST determines how data is justified and bit filled before being passed to the data receive registers (DRR1, DRR2). RJUST is ignored if you enable a companding mode with the RCOMPAND bits. In a companding mode, the 8-bit compressed data in RBR1 is expanded to left-justified 16-bit data in DRR1.<br>Reset type: SYSRSn<br>0h (R/W) = Right justify the data and zero fill the MSBs<br>1h (R/W) = Right justify the data and sign-extend the data into the MSBs<br>2h (R/W) = Left justify the data and zero fill the LSBs<br>3h (R/W) = Reserved (do not use) |



Copyright © 2024 Texas Instruments Incorporated

**Table 21-84. SPCR1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 12-11 | CLKSTP | R/W | 0h | Clock stop mode bits.<br>CLKSTP allows you to use the clock stop mode to support the SPI masterslave protocol. If you will not be using the SPI protocol, you can clear CLKSTP to disable the clock stop mode.<br>In the clock stop mode, the clock stops at the end of each data transfer. At the beginning of eachdata transfer, the clock starts immediately (CLKSTP = 10b) or after a half-cycle delay (CLKSTP = 11b).<br>Reset type: SYSRSn<br>0h (R/W) = Clock stop mode is disabled.<br>1h (R/W) = Clock stop mode is disabled.<br>2h (R/W) = Clock stop mode, without clock delay<br>3h (R/W) = Clock stop mode, with half-cycle clock delay |
| 10-8 | RESERVED | R | 0h | Reserved |
| 7 | DXENA | R/W | 0h | DX delay enabler mode bit.<br>DXENA controls the delay enabler for the DX pin. The enabler creates an extra delay for turn-on time (for the length of the delay, see the device-specific data sheet).<br>Reset type: SYSRSn<br>0h (R/W) = DX delay enabler off<br>1h (R/W) = DX delay enabler on |
| 6 | RESERVED | R/W | 0h | Reserved |
| 5-4 | RINTM | R/W | 0h | Receive interrupt mode bits.<br>RINTM determines which event in the McBSP receiver generates areceive interrupt (RINT) request. If RINT is properly enabled inside the CPU, the CPU services the interrupt request<br>otherwise, the CPU ignores the request.<br>Reset type: SYSRSn<br>0h (R/W) = The McBSP sends a receive interrupt (RINT) request to the CPU when the RRDY bit changes from 0 to 1, indicating that receive data is ready to be read (the content of RBR[1,2] has been copied to DRR[1,2]):<br>Regardless of the value of RINTM, you can check RRDY to determine whether a word transfer is complete.<br>The McBSP sends a RINT request to the CPU when 16 enabled bits have been received on the DR pin.<br>1h (R/W) = In the multichannel selection mode, the McBSP sends a RINT request to the CPU after every 16- channel block is received in a frame.<br>Outside of the multichannel selection mode, no interrupt request is sent.<br>2h (R/W) = The McBSP sends a RINT request to the CPU when each receive frame-synchronization pulse is detected. The interrupt request is sent even if the receiver is in its reset state.<br>3h (R/W) = The McBSP sends a RINT request to the CPU when the RSYNCERR bit is set, indicating a receive frame-synchronization error.<br>Regardless of the value of RINTM, you can check RSYNCERR to determine whether a receive frame-synchronization error occurred. |
| 3 | RSYNCERR | R/W | 0h | Receive frame-sync error bit.<br>RSYNCERR is set when a receive frame-sync error is detected by the McBSP. If RINTM = 11b, the McBSP sends a receive interrupt (RINT) request to the CPU when RSYNCERR is set. The flag remains set until you write a 0 to it or reset the receiver.<br>Reset type: SYSRSn<br>0h (R/W) = No error<br>1h (R/W) = Receive frame-synchronization error. |



Copyright © 2024 Texas Instruments Incorporated

**Table 21-84. SPCR1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 2 | RFULL | R | 0h | Receiver full bit.<br>RFULL is set when the receiver is full with new data and the previously received data has not been read (receiver-full condition). For more details about this condition,<br>Reset type: SYSRSn<br>0h (R/W) = No receiver-full condition<br>1h (R/W) = Receiver-full condition: RSR[1,2] and RBR[1,2] are full with new data, but the previous data in DRR[1,2] has not been read |
| 1 | RRDY | R | 0h | Receiver ready bit.<br>RRDY is set when data is ready to be read from DRR[1,2]. Specifically, RRDY is set in response to a copy from RBR1 to DRR1. If the receive interrupt mode is RINTM = 00b, the McBSP sends a receive interrupt request to the CPU when RRDY changes from 0 to 1.<br>Also, when RRDY changes from 0 to 1, the McBSP sends a receive synchronization event (REVT) signal to the DMA controller.<br>Reset type: SYSRSn<br>0h (R/W) = Receiver not ready<br>When the content of DRR1 is read, RRDY is automatically cleared.<br>1h (R/W) = Receiver ready: New data can be read from DRR[1,2].<br>Important: If both DRRs are required (word length larger than 16 bits), the CPU or the DMA controller must read from DRR2 first and then from DRR1. As soon as DRR1 is read, the next RBR-to-DRR copy is initiated. If DRR2 is not read first, the data in DRR2 is lost. |
| 0 | RRST | R/W | 0h | Receiver reset bit.<br>You can use RRST to take the McBSP receiver into and out of its reset state. This bit has a negative polarity<br>RRST = 0 indicates the reset state.<br>Reset type: SYSRSn<br>0h (R/W) = If you read a 0, the receiver is in its reset state.<br>If you write a 0, you reset the receiver.<br>1h (R/W) = If you read a 1, the receiver is enabled.<br>If you write a 1, you enable the receiver by taking it out of its reset state. |



Copyright © 2024 Texas Instruments Incorporated

### 21.16.2.7 RCR2 Register (Offset = 6h) [Reset = 0000h]

RCR2 is shown in Figure 21-73 and described in Table 21-85.

Return to the Summary Table.

RCR2 contains control bits for the receiver such as number of phases in each frame, the serial word length and number of words for phase 2 of dual phase frames, receive companding mode, receive frame synchronization ignore function, and the receive data delay.

**Figure 21-73. RCR2 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RPHASE | RFRLEN2 | | | | | | |
| R/W-0h | R/W-0h | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RWDLEN2 | | | RCOMPAND | | RFIG | RDATDLY | |
| R/W-0h | | | R/W-0h | | R/W-0h | R/W-0h | |

**Table 21-85. RCR2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RPHASE | R/W | 0h | Receive phase number bit.<br>RPHASE determines whether the receive frame has one phase or two phases. For each phase you can define the serial word length and the number of serial words in the phase. To set up phase 1, program RWDLEN1 (word length) and RFRLEN1 (number of words). To set up phase 2 (if there are two phases), program RWDLEN2 and RFRLEN2.<br>Reset type: SYSRSn<br>0h (R/W) = Single-phase frame<br>The receive frame has only one phase, phase 1.<br>1h (R/W) = Dual-phase frame<br>The receive frame has two phases, phase 1 and phase 2. |
| 14-8 | RFRLEN2 | R/W | 0h | Receive frame length 2 (1 to 128 words). Each frame of receive data can have one or two phases,<br>depending on value that you load into the RPHASE bit. If a single-phase frame is selected, RFRLEN1 in RCR1 selects the number of serial words (8, 12, 16, 20, 24, or 32 bits per word) in the frame. If a dual-phase frame is selected, RFRLEN1 determines the number of serial words in phase 1 of the frame, and RFRLEN2 in RCR2 determines the number of words in phase 2 of the frame. The 7-bit RFRLEN fields allow up to 128 words per phase. See Table 15-77 for a summary of how to determine the frame length. This length corresponds to the number of words or logical time slots or channels per frame-synchronization period.<br>Program the RFRLEN fields with [w minus 1], where w represents the number of words per phase. For example, if you want a phase length of 128 words in phase 2, load 127 into RFRLEN2.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 21-85. RCR2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 7-5 | RWDLEN2 | R/W | 0h | Receive word length 2. Each frame of receive data can have one or two phases, depending on the<br>value that you load into the RPHASE bit. If a single-phase frame is selected, RWDLEN1 in RCR1<br>selects the length for every serial word received in the frame. If a dual-phase frame is selected,<br>RWDLEN1 determines the length of the serial words in phase 1 of the frame, and RWDLEN2 in RCR2<br>determines the word length in phase 2 of the frame.<br>Reset type: SYSRSn<br>0h (R/W) = 8 bits<br>1h (R/W) = 12 bits<br>2h (R/W) = 16 bits<br>3h (R/W) = 20 bits<br>4h (R/W) = 24 bits<br>5h (R/W) = 32 bits<br>6h (R/W) = Reserved (do not use)<br>7h (R/W) = Reserved (do not use) |
| 4-3 | RCOMPAND | R/W | 0h | Receive companding mode bits.<br>Companding (COMpress and exPAND) hardware allows compression and expansion of data in either u-law or A-law format. RCOMPAND allows you to choose one of the following companding modes for the McBSP receiver:<br>For more details about these companding modes, see Section 15.1.5, Companding (Compressing and Expanding) Data.<br>Reset type: SYSRSn<br>0h (R/W) = No companding, any size data, MSB received first<br>1h (R/W) = No companding, 8-bit data, LSB received first<br>2h (R/W) = u-law companding, 8-bit data, MSB received first<br>3h (R/W) = A-law companding, 8-bit data, MSB received first |
| 2 | RFIG | R/W | 0h | Receive frame-synchronization ignore bit. If a frame-synchronization pulse starts the transfer of a new frame before the<br>current frame is fully received, this pulse is treated as an unexpected framesynchronization pulse. Unexpected Receive Frame-Synchronization Pulse.<br>Setting RFIG causes the serial port to ignore unexpected frame-synchronization signals during reception.<br>Reset type: SYSRSn<br>0h (R/W) = Frame-synchronization detect. An unexpected FSR pulse causes the receiver to discard the contents of RSR[1,2] in favor of the new incoming data. The receiver:<br>1. Aborts the current data transfer<br>2. Sets RSYNCERR in SPCR1<br>3. Begins the transfer of a new data word<br>1h (R/W) = Frame-synchronization ignore. An unexpected FSR pulse is ignored. Reception continues uninterrupted. |
| 1-0 | RDATDLY | R/W | 0h | Receive data delay bits.<br>RDATDLY specifies a data delay of 0, 1, or 2 receive clock cycles after framesynchronization and before the reception of the first bit of the frame.<br>Reset type: SYSRSn<br>0h (R/W) = 0-bit data delay<br>1h (R/W) = 1-bit data delay<br>2h (R/W) = 2-bit data delay<br>3h (R/W) = Reserved (do not use) |



Copyright © 2024 Texas Instruments Incorporated

### 21.16.2.8 RCR1 Register (Offset = 7h) [Reset = 0000h]

RCR1 is shown in Figure 21-74 and described in Table 21-86.

Return to the Summary Table.

RCR1 contains control bits for the receiver such as the serial word length and number of words for single phase transmissions, or phase 1 if dual phase frames are used.

**Figure 21-74. RCR1 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | RFRLEN1 | | | | | | |
| R-0h | R/W-0h | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RWDLEN1 | | | RESERVED | | | | |
| R/W-0h | | | R-0h | | | | |

**Table 21-86. RCR1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RESERVED | R | 0h | Reserved |
| 14-8 | RFRLEN1 | R/W | 0h | Receive frame length 1 (1 to 128 words).<br>Each frame of receive data can have one or two phases,depending on value that you load into the RPHASE bit. If a single-phase frame is selected, RFRLEN1 inRCR1 selects the number of serial words (8, 12, 16, 20, 24, or 32 bits per word) in the frame. If a dualphase frame is selected, RFRLEN1 determines the number of serial words in phase 1 of the frame, and RFRLEN2 in RCR2 determines the number of words in phase 2 of the frame. The 7-bit RFRLEN fields allow up to 128 words per phase. See Table 15-75 for a summary of how you determine the frame length. This length corresponds to the number of words or logical time slots or channels per framesynchronization period.<br>Program the RFRLEN fields with [w minus 1], where w represents the number of words per phase. For example, if you want a phase length of 128 words in phase 1, load 127 into RFRLEN1.<br>Note: When operating in SPI mode, the frame length can only be 1 word.<br>Reset type: SYSRSn |
| 7-5 | RWDLEN1 | R/W | 0h | Receive word length 1.<br>Each frame of receive data can have one or two phases, depending on the value that you load into the RPHASE bit. If a single-phase frame is selected, RWDLEN1 in RCR1 selects the length for every serial word received in the frame. If a dual-phase frame is selected, RWDLEN1 determines the length of the serial words in phase 1 of the frame, and RWDLEN2 in RCR2 determines the word length in phase 2 of the frame.<br>Reset type: SYSRSn<br>0h (R/W) = 8 bits<br>1h (R/W) = 12 bits<br>2h (R/W) = 16 bits<br>3h (R/W) = 20 bits<br>4h (R/W) = 24 bits<br>5h (R/W) = 32 bits<br>6h (R/W) = Reserved (do not use)<br>7h (R/W) = Reserved (do not use) |
| 4-0 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 21.16.2.9 XCR2 Register (Offset = 8h) [Reset = 0000h]

XCR2 is shown in Figure 21-75 and described in Table 21-87.

Return to the Summary Table.

XCR2 contains control bits for the transmitter such as number of phases in each frame, the serial word length and number of words for phase 2 of dual phase frames, transmit companding mode, transmit frame synchronization ignore function, and the transmit data delay control.

**Figure 21-75. XCR2 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| XPHASE | XFRLEN2 | | | | | | |
| R/W-0h | R/W-0h | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| XWDLEN2 | | | XCOMPAND | | XFIG | XDATDLY | |
| R/W-0h | | | R/W-0h | | R/W-0h | R/W-0h | |

**Table 21-87. XCR2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | XPHASE | R/W | 0h | Transmit phase number bit.<br>XPHASE determines whether the transmit frame has one phase or two phases. For each phase you can define the serial word length and the number of serial words in the phase. To set up phase 1, program XWDLEN1 (word length) and XFRLEN1 (number of words). To set up phase 2 (if there are two phases), program XWDLEN2 and XFRLEN2.<br>Reset type: SYSRSn |
| 14-8 | XFRLEN2 | R/W | 0h | Transmit frame length 2 (1 to 128 words).<br>Each frame of transmit data can have one or two phases, depending on value that you load into the XPHASE bit. If a single-phase frame is selected, XFRLEN1 in XCR1 selects the number of serial words (8, 12, 16, 20, 24, or 32 bits per word) in the frame. If a dualphase frame is selected, XFRLEN1 determines the number of serial words in phase 1 of the frame and XFRLEN2 in XCR2 determines the number of words in phase 2 of the frame. The 7-bit XFRLEN fields allow up to 128 words per phase. See Table 15-81 for a summary of how to determine the frame length. This length corresponds to the number of words or logical time slots or channels per framesynchronization period.<br>Program the XFRLEN fields with [w minus 1], where w represents the number of words per phase. For example, if you want a phase length of 128 words in phase 1, load 127 into XFRLEN1.<br>Reset type: SYSRSn |
| 7-5 | XWDLEN2 | R/W | 0h | Transmit word length 2.<br>Each frame of transmit data can have one or two phases, depending on the value that you load into the XPHASE bit. If a single-phase frame is selected, XWDLEN1 in XCR1 selects the length for every serial word transmitted in the frame. If a dual-phase frame is selected, XWDLEN1 determines the length of the serial words in phase 1 of the frame and XWDLEN2 in XCR2 determines the word length in phase 2 of the frame.<br>Reset type: SYSRSn<br>0h (R/W) = 8 bits<br>1h (R/W) = 12 bits<br>2h (R/W) = 16 bits<br>3h (R/W) = 20 bits<br>4h (R/W) = 24 bits<br>5h (R/W) = 32 bits<br>6h (R/W) = Reserved (do not use)<br>7h (R/W) = Reserved (do not use) |



Copyright © 2024 Texas Instruments Incorporated

**Table 21-87. XCR2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 4-3 | XCOMPAND | R/W | 0h | Transmit companding mode bits.<br>Companding (COMpress and exPAND) hardware allows compression and expansion of data in either u-law or A-law format.<br>Reset type: SYSRSn<br>0h (R/W) = No companding, any size data, MSB received first<br>1h (R/W) = No companding, 8-bit data, LSB received first<br>2h (R/W) = u-law companding, 8-bit data, MSB received first<br>3h (R/W) = A-law companding, 8-bit data, MSB received first |
| 2 | XFIG | R/W | 0h | Transmit frame-synchronization ignore bit.<br>If a frame-synchronization pulse starts the transfer of a new frame before the current frame is fully transmitted, this pulse is treated as an unexpected framesynchronization pulse.<br>Setting XFIG causes the serial port to ignore unexpected frame-synchronization pulses during transmission.<br>Reset type: SYSRSn<br>0h (R/W) = Frame-synchronization detect. An unexpected FSX pulse causes the transmitter to discard the content of XSR[1,2]. The transmitter:<br>1. Aborts the present transmission<br>2. Sets XSYNCERR in SPCR2<br>3. Begins a new transmission from DXR[1,2]. If new data was written to DXR[1,2] since the last DXR[1,2]-to-XSR[1,2] copy, the current value in XSR[1,2] is lost. Otherwise, the same data is transmitted.<br>1h (R/W) = Frame-synchronization ignore. An unexpected FSX pulse is ignored. Transmission continues uninterrupted. |
| 1-0 | XDATDLY | R/W | 0h | Transmit data delay bits. XDATDLY specifies a data delay of 0, 1, or 2 transmit clock cycles after frame synchronization and before the transmission of the first bit of the frame.<br>Reset type: SYSRSn<br>0h (R/W) = 0-bit data delay<br>1h (R/W) = 1-bit data delay<br>2h (R/W) = 2-bit data delay<br>3h (R/W) = Reserved (do not use) |



Copyright © 2024 Texas Instruments Incorporated

### 21.16.2.10 XCR1 Register (Offset = 9h) [Reset = 0000h]

XCR1 is shown in Figure 21-76 and described in Table 21-88.

Return to the Summary Table.

XCR1 contains control bits for the transmitter such as the serial word length and number of words for single phase transmissions, or phase 1 if dual phase frames are used.

**Figure 21-76. XCR1 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | XFRLEN1 | | | | | | |
| R-0h | R/W-0h | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| XWDLEN1 | | | RESERVED | | | | |
| R/W-0h | | | R-0h | | | | |

**Table 21-88. XCR1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RESERVED | R | 0h | Reserved |
| 14-8 | XFRLEN1 | R/W | 0h | Transmit frame length 1 (1 to 128 words). Each frame of transmit data can have one or two phases, depending on value that you load into the XPHASE bit. If a single-phase frame is selected, XFRLEN1 in XCR1 selects the number of serial words (8, 12, 16, 20, 24, or 32 bits per word) in the frame. If a dual-phase frame is selected, XFRLEN1 determines the number of serial words in phase 1 of the frame and XFRLEN2 in XCR2 determines the number of words in phase 2 of the frame. The 7-bit XFRLEN fields allow up to 128 words per phase. See Table 15-79 for a summary of how you determine the frame length. This length corresponds to the number of words or logical time slots or channels per frame-synchronization period. Program the XFRLEN fields with [w minus 1], where w represents the number of words per phase. For example, if you want a phase length of 128 words in phase 1, load 127 into XFRLEN1.<br>Note: When operating in SPI mode, the frame length can only be 1 word.<br>Reset type: SYSRSn |
| 7-5 | XWDLEN1 | R/W | 0h | Transmit word length 1.<br>Each frame of transmit data can have one or two phases, depending on the value that you load into the XPHASE bit. If a single-phase frame is selected, XWDLEN1 in XCR1 selects the length for every serial word transmitted in the frame. If a dual-phase frame is selected, XWDLEN1 determines the length of the serial words in phase 1 of the frame and XWDLEN2 in XCR2 determines the word length in phase 2 of the frame.<br>Reset type: SYSRSn<br>0h (R/W) = 8 bits<br>1h (R/W) = 12 bits<br>2h (R/W) = 16 bits<br>3h (R/W) = 20 bits<br>4h (R/W) = 24 bits<br>5h (R/W) = 32 bits<br>6h (R/W) = Reserved (do not use)<br>7h (R/W) = Reserved (do not use) |
| 4-0 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 21.16.2.11 SRGR2 Register (Offset = Ah) [Reset = 0000h]

SRGR2 is shown in Figure 21-77 and described in Table 21-89.

Return to the Summary Table.

SRGR2 contains control bits for the sample rate generator such as input clock selection, internal tranmist frame-synchronization source selection, and the period between frame-synchronization pulses.
If an external source provides the input clock source for the sample rate generator, a control bit is provided to make the CLKG synchronized to an external frame synchronization pulse on the FSR pin so that CLKG is kept in phase with the input clock.

**Figure 21-77. SRGR2 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| GSYNC | RESERVED | CLKSM | FSGM | FPER | | | |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| FPER | | | | | | | |
| R/W-0h | | | | | | | |

**Table 21-89. SRGR2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | GSYNC | R/W | 0h | Clock synchronization mode bit for CLKG.<br>GSYNC is used only when the input clock source for the sample rate generator is external?on the MCLKR pin.<br>When GSYNC = 1, the clock signal (CLKG) and the frame-synchronization signal (FSG) generated by the sample rate generator are made dependent on pulses on the FSR pin.<br>Reset type: SYSRSn<br>0h (R/W) = No clock synchronization<br>CLKG oscillates without adjustment, and FSG pulses every (FPER + 1) CLKG cycles.<br>1h (R/W) = Clock synchronization<br>- CLKG is adjusted as necessary so that it is synchronized with the input clock on the<br>MCLKR pin.<br>- FSG pulses. FSG only pulses in response to a pulse on the FSR pin.<br>The frame-synchronization period defined in FPER is ignored. |
| 14 | RESERVED | R/W | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 21-89. SRGR2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 13 | CLKSM | R/W | 0h | Sample rate generator mode<br>Reset type: SYSRSn<br>0h (R/W) = Sample rate generator input clock mode bit. The sample rate generator can accept an input clock signal and divide it down according to CLKGDV to produce an output clock signal, CLKG. The frequency of CLKG is:)<br>CLKG frequency = (input clock frequency)/ (CLKGDV + 1<br>CLKSM is used in conjunction with the SCLKME bit to determine the source for the input clock.<br>A reset selects the CPU clock as the input clock and forces the CLKG frequency to half the LSPCLK frequency.<br>The input clock for the sample rate generator is taken from the MCLKR pin, depending on the value of the SCLKME bit of PCR:<br>SCLKME CLKSM Input Clock For<br>Sample Rate Generator<br>0 0 Reserved<br>1 0 Signal on MCLKR pin<br>1h (R/W) = The input clock for the sample rate generator is taken from the LSPCLK or from the MCLKX pin, depending on the value of the SCLKME bit of PCR:<br>SCLKME CLKSM Input Clock For<br>Sample Rate Generator<br>0 1 LSPCLK<br>1 1 Signal on MCLKX pin |
| 12 | FSGM | R/W | 0h | Sample rate generator transmit frame-synchronization mode bit. The transmitter can get frame synchronization from the FSX pin (FSXM = 0) or from inside the McBSP (FSXM = 1). When FSXM = 1, the FSGM bit determines how the McBSP supplies frame-synchronization pulses.<br>Reset type: SYSRSn<br>0h (R/W) = If FSXM = 1, the McBSP generates a transmit frame-synchronization pulse when the content of DXR[1,2] is copied to XSR[1,2].<br>1h (R/W) = If FSXM = 1, the transmitter uses frame-synchronization pulses generated by the sample rate generator. Program the FWID bits to set the width of each pulse. Program the FPER bits to set the period between pulses. |
| 11-0 | FPER | R/W | 0h | Frame-synchronization period bits for FSG.<br>The sample rate generator can produce a clock signal, CLKG, and a frame-synchronization signal, FSG. The period between framesynchronization pulses on FSG is (FPER + 1) CLKG cycles. The 12 bits of FPER allow a frame-synchronization period of 1 to 4096 CLKG cycles:<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 21.16.2.12 SRGR1 Register (Offset = Bh) [Reset = 0001h]

SRGR1 is shown in Figure 21-78 and described in Table 21-90.

Return to the Summary Table.

SRGR1 contains control bits for the sample rate generator functions such as the divide down frequency, and the width fo the frame-synchronization pulses on FSG.

**Figure 21-78. SRGR1 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| FWID | | | | | | | |
| R/W-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CLKGDV | | | | | | | |
| R/W-1h | | | | | | | |

**Table 21-90. SRGR1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | FWID | R/W | 0h | Divide-down value for CLKG.<br>The sample rate generator can accept an input clock signal and divide it down according to CLKGDV to produce an output clock signal, CLKG. The frequency of CLKG is:<br>CLKG frequency = (Input clock frequency)/ (CLKGDV + 1)<br>The input clock is selected by the SCLKME and CLKSM bits:<br>SCLKME CLKSM Input Clock For<br>Sample Rate Generator<br>0 0 Reserved<br>0 1 LSPCLK<br>1 0 Signal on MCLKR pin<br>1 1 Signal on MCLKX pin<br>Reset type: SYSRSn |
| 7-0 | CLKGDV | R/W | 1h | Frame-synchronization pulse width bits for FSG<br>The sample rate generator can produce a clock signal, CLKG, and a frame-synchronization signal, FSG. For frame-synchronization pulses on FSG, (FWID + 1) is the pulse width in CLKG cycles. The eight bits of FWID allow a pulse width of 1 to 256 CLKG cycles:<br>0 <= FWID <= 255<br>1 <= (FWID + 1) <= 256 CLKG cycles<br>The period between the frame-synchronization pulses on FSG is defined by the FPER bits.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 21.16.2.13 MCR2 Register (Offset = Ch) [Reset = 0000h]

MCR2 is shown in Figure 21-79 and described in Table 21-91.

Return to the Summary Table.

MCR2 contains control bits for the transmitter multi-channel functions such as channel enable mode selection, channel partition modes, channel block assignments, and active channel status bits.

**Figure 21-79. MCR2 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | XMCME | XPBBLK |
| R-0h | | | | | | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| XPBBLK | XPABLK | | XCBLK | | | XMCM | |
| R/W-0h | R/W-0h | | R-0h | | | R/W-0h | |

**Table 21-91. MCR2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-10 | RESERVED | R | 0h | Reserved |
| 9 | XMCME | R/W | 0h | Transmit multichannel partition mode bit.<br>XMCME determines whether only 32 channels or all 128 channels are to be individually selectable. XMCME is only applicable if channels can be individually disabled/enabled or masked/unmasked for transmission (XMCM is nonzero).<br>Reset type: SYSRSn |
| 8-7 | XPBBLK | R/W | 0h | Transmit partition B block bits<br>XPBBLK is only applicable if channels can be individually disabled/ enabled and masked/unmasked (XMCM is nonzero) and the 2-partition mode is selected (XMCME = 0). Under these conditions, the McBSP transmitter can transmit or withhold data in any of the 32 channels that are assigned to<br>partitions A and B of the transmitter.<br>The 128 transmit channels of the McBSP are divided equally among 8 blocks (0 through 7). When XPBBLK is applicable, use XPBBLK to assign one of the odd-numbered blocks (1, 3, 5, or 7) to partition B, as shown in the following table. Use the PABLK bit to assign one of the evennumbered blocks (0, 2, 4, or 6) to partition A.<br>If you want to use more than 32 channels, you can change block assignments dynamically. You can assign a new block to one partition while the transmitter is handling activity in the other partition. For example, while the block in partition A is active, you can change which block is assigned to partition B. The XCBLK bits are regularly pdated to indicate which block is active.<br>When XMCM = 11b (for symmetric transmission and reception), the transmitter uses the receive block bits (RPABLK and RPBBLK) rather than the transmit block bits (XPABLK and XPBBLK).<br>Reset type: SYSRSn<br>0h (R/W) = Block 1: channels 16 through 31<br>1h (R/W) = Block 3: channels 48 through 63<br>2h (R/W) = Block 5: channels 80 through 95<br>3h (R/W) = Block 7: channels 112 through 127 |



Copyright © 2024 Texas Instruments Incorporated

**Table 21-91. MCR2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 6-5 | XPABLK | R/W | 0h | Transmit partition A block bits.<br>XPABLK is only applicable if channels can be individually disabled/enabled and masked/unmasked (XMCM is nonzero) and the 2-partition mode is selected (XMCME = 0). Under these conditions, the McBSP transmitter can transmit or withhold data in any of the 32 channels that are assigned to artitions A and B of the transmitter. See the Descriptionfor XPBBLK (bits 8-7) for more information about assigning blocks to partitions A and B.<br>Reset type: SYSRSn<br>0h (R/W) = Block 0: channels 0 through 15<br>1h (R/W) = Block 2: channels 32 through 47<br>2h (R/W) = Block 4: channels 64 through 79<br>3h (R/W) = Block 6: channels 96 through 111 |
| 4-2 | XCBLK | R | 0h | Transmit current block indicator.<br>XCBLK indicates which block of 16 channels is involved in the current McBSP transmission:<br>Reset type: SYSRSn<br>0h (R/W) = Block 0: channels 0 through 15<br>1h (R/W) = Block 1: channels 16 through 31<br>2h (R/W) = Block 2: channels 32 through 47<br>3h (R/W) = Block 3: channels 48 through 63<br>4h (R/W) = Block 4: channels 64 through 79<br>5h (R/W) = Block 5: channels 80 through 95<br>6h (R/W) = Block 6: channels 96 through 111<br>7h (R/W) = Block 7: channels 112 through 127 |
| 1-0 | XMCM | R/W | 0h | Transmit multichannel selection mode bits.<br>XMCM determines whether all channels or only selected channels are enabled and unmasked for transmission.<br>Reset type: SYSRSn<br>0h (R/W) = No transmit multichannel selection mode is on. All channels are enabled and unmasked. No channels can be disabled or masked<br>1h (R/W) = All channels are disabled unless they are selected in the appropriate transmit channel enable registers (XCERs). If enabled, a channel in this mode is also unmasked.<br>The XMCME bit determines whether 32 channels or 128 channels are selectable in XCERs.<br>2h (R/W) = All channels are enabled, but they are masked unless they are selected in the appropriate transmit channel enable registers (XCERs).<br>The XMCME bit determines whether 32 channels or 128 channels are selectable in XCERs.<br>3h (R/W) = This mode is used for symmetric transmission and reception.<br>All channels are disabled for transmission unless they are enabled for reception in the appropriate receive channel enable registers (RCERs). Once enabled, they are masked unless they are also selected in the appropriate transmit channel enable registers (XCERs).<br>The XMCME bit determines whether 32 channels or 128 channels are selectable in RCERs and XCERs. |



Copyright © 2024 Texas Instruments Incorporated

### 21.16.2.14 MCR1 Register (Offset = Dh) [Reset = 0000h]

MCR1 is shown in Figure 21-80 and described in Table 21-92.

Return to the Summary Table.

MCR1 contains control bits for the receiver multi-channel functions such as channel enable mode selection, channel partition modes, channel block assignments, and active channel status bits.

**Figure 21-80. MCR1 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | RMCME | RPBBLK |
| R-0h | | | | | | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RPBBLK | RPABLK | | RCBLK | | | RESERVED | RMCM |
| R/W-0h | R/W-0h | | R-0h | | | R/W-0h | R/W-0h |

**Table 21-92. MCR1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-10 | RESERVED | R | 0h | Reserved |
| 9 | RMCME | R/W | 0h | Receive multichannel partition mode bit.<br>RMCME is only applicable if channels can be individually enabled or disabled for reception (RMCM = 1).<br>RMCME determines whether only 32 channels or all 128 channels are to be individually selectable.<br>Reset type: SYSRSn<br>0h (R/W) = 2-partition mode<br>Only partitions A and B are used. You can control up to 32 channels in the receive multichannel selection mode (RMCM = 1).<br>Assign 16 channels to partition A with the RPABLK bits.<br>Assign 16 channels to partition B with the RPBBLK bits.<br>You control the channels with the appropriate receive channel enable registers:<br>RCERA: Channels in partition A<br>RCERB: Channels in partition B<br>1h (R/W) = 8-partition mode<br>All partitions (A through H) are used. You can control up to 128 channels in the receive multichannel selection mode. You control the channels with the appropriate receive channel enable registers:<br>RCERA: Channels 0 through 15<br>RCERB: Channels 16 through 31<br>RCERC: Channels 32 through 47<br>RCERD: Channels 48 through 63<br>RCERE: Channels 64 through 79<br>RCERF: Channels 80 through 95<br>RCERG: Channels 96 through 111<br>RCERH: Channels 112 through 127 |



Copyright © 2024 Texas Instruments Incorporated

**Table 21-92. MCR1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 8-7 | RPBBLK | R/W | 0h | Receive partition B block bits<br>RPBBLK is only applicable if channels can be individually enabled or disabled (RMCM = 1) and the 2-partition mode is selected (RMCME = 0). Under these conditions, the McBSP receiver can accept or ignore data in any of the 32 channels that are assigned to partitions A and B of the receiver.<br>The 128 receive channels of the McBSP are divided equally among 8 blocks (0 through 7). When RPBBLK is applicable, use RPBBLK to assign one of the odd-numbered blocks (1, 3, 5, or 7) to partition B. Use the RPABLK bits to assign one of the even-numbered blocks (0, 2, 4, or 6) to partition A.<br>If you want to use more than 32 channels, you can change block assignments dynamically. You can assign a new block to one partition while the receiver is handling activity in the other partition. For example, while the block in partition A is active, you can change which block is assigned to partition B. The RCBLK bits are regularly updated to indicate which block is active.<br>When XMCM = 11b (for symmetric transmission and reception), the transmitter uses the receive block bits (RPABLK and RPBBLK) rather than the transmit block bits (XPABLK and XPBBLK).<br>Reset type: SYSRSn<br>0h (R/W) = Block 1: channels 16 through 31<br>1h (R/W) = Block 3: channels 48 through 63<br>2h (R/W) = Block 5: channels 80 through 95<br>3h (R/W) = Block 7: channels 112 through 127 |
| 6-5 | RPABLK | R/W | 0h | Receive partition A block bits<br>RPABLK is only applicable if channels can be individually enabled or disabled (RMCM = 1) and the 2-partition mode is selected (RMCME = 0). Under these conditions, the McBSP receiver can accept or ignore data in any of the 32 channels that are assigned to partitions A and B of the receiver. See the Descriptionfor RPBBLK (bits 8-7) for more information about assigning blocks to partitions A and B.<br>Reset type: SYSRSn<br>0h (R/W) = Block 0: channels 0 through 15<br>1h (R/W) = Block 2: channels 32 through 47<br>2h (R/W) = Block 4: channels 64 through 79<br>3h (R/W) = Block 6: channels 96 through 111 |
| 4-2 | RCBLK | R | 0h | Receive current block indicator.<br>RCBLK indicates which block fo 16 channels is involved in the current McBSP reception<br>Reset type: SYSRSn<br>0h (R/W) = Block 0: channels 0 through 15<br>1h (R/W) = Block 1: channels 16 through 31<br>2h (R/W) = Block 2: channels 32 through 47<br>3h (R/W) = Block 3: channels 48 through 63<br>4h (R/W) = Block 4: channels 64 through 79<br>5h (R/W) = Block 5: channels 80 through 95<br>6h (R/W) = Block 6: channels 96 through 111<br>7h (R/W) = Block 7: channels 112 through 127 |
| 1 | RESERVED | R/W | 0h | Reserved |
| 0 | RMCM | R/W | 0h | Receive multichannel selection mode bit. RMCM determines whether all channels or only selected channels are enabled for reception:<br>Reset type: SYSRSn<br>0h (R/W) = All 128 channels are enabled.<br>1h (R/W) = Multichanneled selection mode. Channels can be individually enabled or disabled.<br>The only channels enabled are those selected in the appropriate receive channel enable registers (RCERs). The way channels are assigned to the RCERs depends on the number of receive channel partitions (2 or 8), as defined by the RMCME bit. |



Copyright © 2024 Texas Instruments Incorporated

### 21.16.2.15 RCERA Register (Offset = Eh) [Reset = 0000h]

RCERA is shown in Figure 21-81 and described in Table 21-93.

Return to the Summary Table.

RCERA contains the receive channel enable registers for the A partition. This register is only used when the receiver is configured to allow individual disabling/enabling and masking/unmasking of the channels (e.g. RMCM is nonzero).

**Figure 21-81. RCERA Register**

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

**Table 21-93. RCERA Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RCEA | R/W | 0h | Receive channel enable bit.<br>For receive multichannel selection mode (RMCM = 1):<br>Reset type: SYSRSn<br>0h (R/W) = Disable the channel that is mapped to RCEx.<br>1h (R/W) = Enable the channel that is mapped to RCEx. |



Copyright © 2024 Texas Instruments Incorporated

### 21.16.2.16 RCERB Register (Offset = Fh) [Reset = 0000h]

RCERB is shown in Figure 21-82 and described in Table 21-94.

Return to the Summary Table.

RCERB contains the receive channel enable registers for the B partition. This register is only used when the receiver is configured to allow individual disabling/enabling and masking/unmasking of the channels (e.g. RMCM is nonzero).

**Figure 21-82. RCERB Register**

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

**Table 21-94. RCERB Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RCEB | R/W | 0h | Receive channel enable bit.<br>For receive multichannel selection mode (RMCM = 1):<br>Reset type: SYSRSn<br>0h (R/W) = Disable the channel that is mapped to RCEx.<br>1h (R/W) = Enable the channel that is mapped to RCEx. |



Copyright © 2024 Texas Instruments Incorporated

### 21.16.2.17 XCERA Register (Offset = 10h) [Reset = 0000h]

XCERA is shown in Figure 21-83 and described in Table 21-95.

Return to the Summary Table.

XCERA contains the transmit channel enable registers for the A partition. This register is only used when the transmitter is configured to allow individual disabling/enabling and masking/unmasking of the channels (e.g. XMCM is nonzero).

**Figure 21-83. XCERA Register**

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

**Table 21-95. XCERA Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | XCERA | R/W | 0h | Transmit channel enable bit.<br>The role of this bit depends on which transmit multichannel selection mode is selected with the XMCM bits.<br>Reset type: SYSRSn<br>0h (R/W) = For multichannel selection when XMCM = 01b (all channels disabled unless selected):<br>Disable and mask the channel that is mapped to XCEx.<br>For multichannel selection when XMCM = 10b (all channels enabled but masked unless selected):<br>Mask the channel that is mapped to XCEx.<br>For multichannel selection when XMCM = 11b (all channels masked unless selected):<br>Mask the channel that is mapped to XCEx. Even if the channel is enabled by the corresponding receive channel enable bit, this channel's data cannot appear on the DX pin.<br>1h (R/W) = For multichannel selection when XMCM = 01b (all channels disabled unless selected):<br>Enable and unmask the channel that is mapped to XCEx.<br>For multichannel selection when XMCM = 10b (all channels enabled but masked unless selected):<br>Unmask the channel that is mapped to XCEx.<br>For multichannel selection when XMCM = 11b (all channels masked unless selected):<br>Unmask the channel that is mapped to XCEx. If the channel is also enabled by the corresponding receive channel enable bit, full transmission can occur. |



Copyright © 2024 Texas Instruments Incorporated

### 21.16.2.18 XCERB Register (Offset = 11h) [Reset = 0000h]

XCERB is shown in Figure 21-84 and described in Table 21-96.

Return to the Summary Table.

XCERB contains the transmit channel enable registers for the B partition. This register is only used when the transmitter is configured to allow individual disabling/enabling and masking/unmasking of the channels (e.g. XMCM is nonzero).

**Figure 21-84. XCERB Register**

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

**Table 21-96. XCERB Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | XCERB | R/W | 0h | Transmit channel enable bit.<br>The role of this bit depends on which transmit multichannel selection mode is selected with the XMCM bits.<br>Reset type: SYSRSn<br>0h (R/W) = For multichannel selection when XMCM = 01b (all channels disabled unless selected):<br>Disable and mask the channel that is mapped to XCEx.<br>For multichannel selection when XMCM = 10b (all channels enabled but masked unless selected):<br>Mask the channel that is mapped to XCEx.<br>For multichannel selection when XMCM = 11b (all channels masked unless selected):<br>Mask the channel that is mapped to XCEx. Even if the channel is enabled by the corresponding receive channel enable bit, this channel's data cannot appear on the DX pin.<br>1h (R/W) = For multichannel selection when XMCM = 01b (all channels disabled unless selected):<br>Enable and unmask the channel that is mapped to XCEx.<br>For multichannel selection when XMCM = 10b (all channels enabled but masked unless selected):<br>Unmask the channel that is mapped to XCEx.<br>For multichannel selection when XMCM = 11b (all channels masked unless selected):<br>Unmask the channel that is mapped to XCEx. If the channel is also enabled by the corresponding receive channel enable bit, full transmission can occur. |



Copyright © 2024 Texas Instruments Incorporated

### 21.16.2.19 PCR Register (Offset = 12h) [Reset = 0000h]

PCR is shown in Figure 21-85 and described in Table 21-97.

Return to the Summary Table.

PCR contains control bits for the McBSP pins such as frame synchronization modes, clock modes, input clock source selection for the sample rate generator, frame synchronization pulse active polarity, and transmit/receive active edge selection.

**Figure 21-85. PCR Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | FSXM | FSRM | CLKXM | CLKRM |
| R-0h | | | | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SCLKME | RESERVED | RESERVED | RESERVED | FSXP | FSRP | CLKXP | CLKRP |
| R/W-0h | R-0h | R-0h | R-0h | R/W-0h | R-0h | R-0h | R/W-0h |

**Table 21-97. PCR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | RESERVED | R | 0h | Reserved |
| 11 | FSXM | R/W | 0h | Transmit frame-synchronization mode bit.<br>FSXM determines whether transmit framesynchronization pulses are supplied externally or internally. The polarity of the signal on the FSX pin is determined by the FSXP bit.<br>Reset type: SYSRSn<br>0h (R/W) = Transmit frame synchronization is supplied by an external source via the FSX pin.<br>1h (R/W) = Transmit frame synchronization is generated internally by the Sample Rate generator, as determined by the FSGM bit of SRGR2 |
| 10 | FSRM | R/W | 0h | Receive frame-synchronization mode bit.<br>FSRM determines whether receive framesynchronization pulses are supplied externally or internally. The polarity of the signal on the FSR pin is determined by the FSRP bit.<br>Reset type: SYSRSn<br>0h (R/W) = Receive frame synchronization is supplied by an external source via the FSR pin.<br>1h (R/W) = Receive frame synchronization is supplied by the sample rate generator. FSR is an output pin reflecting internal FSR, except when GSYNC = 1 in SRGR2 |



Copyright © 2024 Texas Instruments Incorporated

**Table 21-97. PCR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 9 | CLKXM | R/W | 0h | Transmit clock mode bit.<br>CLKXM determines whether the source for the transmit clock is external or internal, and whether the MCLKX pin is an input or an output. The polarity of the signal on the MCLKX pin is determined by the CLKXP bit.<br>In the clock stop mode (CLKSTP = 10b or 11b), the McBSP can act as a master or as a slave in the SPI protocol. If the McBSP is a master, make sure that CLKX is an output. If the McBSP is a slave, make sure that CLKX is an input.<br>Reset type: SYSRSn<br>0h (R/W) = Not in clock stop mode (CLKSTP = 00b or 01b):<br>The transmitter gets its clock signal from an external source via the MCLKX pin.<br>In clock stop mode (CLKSTP = 10b or 11b):<br>The McBSP is a slave in the SPI protocol. The internal transmit clock (CLKX) is driven by the SPI master via the MCLKX pin. The internal receive clock (MCLKR) is driven internally by CLKX, so that both the transmitter and the receiver are controlled by the external master clock.<br>1h (R/W) = Not in clock stop mode (CLKSTP = 00b or 01b):<br>Internal CLKX is driven by the sample rate generator of the McBSP. The MCLKX pin is an output pin that reflects internal CLKX.<br>In clock stop mode (CLKSTP = 10b or 11b):<br>The McBSP is a master in the SPI protocol. The sample rate generator drives the internal transmit clock (CLKX). Internal CLKX is reflected on the MCLKX pin to drive the shift clock of the SPI-compliant slaves in the system. Internal CLKX also drives the internal receive clock (MCLKR), so that both the transmitter and the receiver are controlled by the internal master clock |
| 8 | CLKRM | R/W | 0h | Receive clock mode bit.<br>The role of CLKRM and the resulting effect on the MCLKR pin depend on whether the McBSP is in the digital loopback mode (DLB = 1).<br>The polarity of the signal on the MCLKR pin is determined by the CLKRP bit.<br>Reset type: SYSRSn<br>0h (R/W) = Not in digital loopback mode (DLB = 0):<br>The MCLKR pin is an input pin that supplies the internal receive clock (MCLKR).<br>In digital loopback mode (DLB = 1):<br>The MCLKR pin is in the high impedance state. The internal receive clock (MCLKR) is driven by the internal transmit clock (CLKX). CLKX is derived according to the CLKXM bit.<br>1h (R/W) = Not in digital loopback mode (DLB = 0):<br>Internal MCLKR is driven by the sample rate generator of the McBSP. The MCLKR pin is an output pin that reflects internal MCLKR.<br>In digital loopback mode (DLB = 1):<br>Internal MCLKR is driven by internal CLKX. The MCLKR pin is an output pin that reflects internal MCLKR. CLKX is derived according to the CLKXM bit. |



Copyright © 2024 Texas Instruments Incorporated

**Table 21-97. PCR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 7 | SCLKME | R/W | 0h | Sample rate generator input clock mode bit. The sample rate generator can produce a clock signal, CLKG. The frequency of CLKG is:<br>CLKG freq. = (Input clock frequency) / (CLKGDV + 1)<br>SCLKME is used in conjunction with the CLKSM bit to select the input clock.<br>SCLKME CLKSM Input Clock For Sample Rate Generator<br>0 0 Reserved<br>0 1 LSPCLK<br>The input clock for the sample rate generator is taken from the MCLKR pin or from the MCLKX pin, depending on the value of the CLKSM bit of SRGR2:<br>SCLKME CLKSM Input Clock For Sample Rate Generator<br>1 0 Signal on MCLKR pin<br>1 1 Signal on MCLKX pin<br>Reset type: SYSRSn |
| 6 | RESERVED | R | 0h | Reserved |
| 5 | RESERVED | R | 0h | Reserved |
| 4 | RESERVED | R | 0h | Reserved |
| 3 | FSXP | R/W | 0h | Transmit frame-synchronization polarity bit. FSXP determines the polarity of FSX as seen on the FSX pin.<br>Reset type: SYSRSn<br>0h (R/W) = Transmit frame-synchronization pulses are active high.<br>1h (R/W) = Transmit frame-synchronization pulses are active low. |
| 2 | FSRP | R | 0h | Receive frame-synchronization polarity bit. FSRP determines the polarity of FSR as seen on the FSR pin.<br>Reset type: SYSRSn<br>0h (R/W) = Receive frame-synchronization pulses are active high.<br>1h (R/W) = Receive frame-synchronization pulses are active low. |
| 1 | CLKXP | R | 0h | Transmit clock polarity bit. CLKXP determines the polarity of CLKX as seen on the MCLKX pin.<br>Reset type: SYSRSn<br>0h (R/W) = Transmit data is sampled on the rising edge of CLKX.<br>1h (R/W) = Transmit data is sampled on the falling edge of CLKX. |
| 0 | CLKRP | R/W | 0h | Receive clock polarity bit. CLKRP determines the polarity of CLKR as seen on the MCLKR pin.<br>Reset type: SYSRSn<br>0h (R/W) = Receive data is sampled on the falling edge of MCLKR.<br>1h (R/W) = Receive data is sampled on the rising edge of MCLKR. |



Copyright © 2024 Texas Instruments Incorporated

### 21.16.2.20 RCERC Register (Offset = 13h) [Reset = 0000h]

RCERC is shown in Figure 21-86 and described in Table 21-98.

Return to the Summary Table.

RCERC contains the receive channel enable registers for the C partition. This register is only used when the receiver is configured to allow individual disabling/enabling and masking/unmasking of the channels (e.g. RMCM is nonzero).

**Figure 21-86. RCERC Register**

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

**Table 21-98. RCERC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RCEC | R/W | 0h | Receive channel enable bit.<br>For receive multichannel selection mode (RMCM = 1):<br>Reset type: SYSRSn<br>0h (R/W) = Disable the channel that is mapped to RCEx.<br>1h (R/W) = Enable the channel that is mapped to RCEx. |



Copyright © 2024 Texas Instruments Incorporated

### 21.16.2.21 RCERD Register (Offset = 14h) [Reset = 0000h]

RCERD is shown in Figure 21-87 and described in Table 21-99.

Return to the Summary Table.

RCERD contains the receive channel enable registers for the D partition. This register is only used when the receiver is configured to allow individual disabling/enabling and masking/unmasking of the channels (e.g. RMCM is nonzero).

**Figure 21-87. RCERD Register**

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

**Table 21-99. RCERD Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RCED | R/W | 0h | Receive channel enable bit.<br>For receive multichannel selection mode (RMCM = 1):<br>Reset type: SYSRSn<br>0h (R/W) = Disable the channel that is mapped to RCEx.<br>1h (R/W) = Enable the channel that is mapped to RCEx. |



Copyright © 2024 Texas Instruments Incorporated

### 21.16.2.22 XCERC Register (Offset = 15h) [Reset = 0000h]

XCERC is shown in Figure 21-88 and described in Table 21-100.

Return to the Summary Table.

XCERC contains the transmit channel enable registers for the C partition. This register is only used when the transmitter is configured to allow individual disabling/enabling and masking/unmasking of the channels (e.g. XMCM is nonzero).

**Figure 21-88. XCERC Register**

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

**Table 21-100. XCERC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | XCERC | R/W | 0h | Transmit channel enable bit.<br>The role of this bit depends on which transmit multichannel selection mode is selected with the XMCM bits.<br>Reset type: SYSRSn<br>0h (R/W) = For multichannel selection when XMCM = 01b (all channels disabled unless selected):<br>Disable and mask the channel that is mapped to XCEx.<br>For multichannel selection when XMCM = 10b (all channels enabled but masked unless selected):<br>Mask the channel that is mapped to XCEx.<br>For multichannel selection when XMCM = 11b (all channels masked unless selected):<br>Mask the channel that is mapped to XCEx. Even if the channel is enabled by the corresponding receive channel enable bit, this channel's data cannot appear on the DX pin.<br>1h (R/W) = For multichannel selection when XMCM = 01b (all channels disabled unless selected):<br>Enable and unmask the channel that is mapped to XCEx.<br>For multichannel selection when XMCM = 10b (all channels enabled but masked unless selected):<br>Unmask the channel that is mapped to XCEx.<br>For multichannel selection when XMCM = 11b (all channels masked unless selected):<br>Unmask the channel that is mapped to XCEx. If the channel is also enabled by the corresponding receive channel enable bit, full transmission can occur. |



Copyright © 2024 Texas Instruments Incorporated

### 21.16.2.23 XCERD Register (Offset = 16h) [Reset = 0000h]

XCERD is shown in Figure 21-89 and described in Table 21-101.

Return to the Summary Table.

XCERD contains the transmit channel enable registers for the D partition. This register is only used when the transmitter is configured to allow individual disabling/enabling and masking/unmasking of the channels (e.g. XMCM is nonzero).

**Figure 21-89. XCERD Register**

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

**Table 21-101. XCERD Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | XCERD | R/W | 0h | Transmit channel enable bit.<br>The role of this bit depends on which transmit multichannel selection mode is selected with the XMCM bits.<br>Reset type: SYSRSn<br>0h (R/W) = For multichannel selection when XMCM = 01b (all channels disabled unless selected):<br>Disable and mask the channel that is mapped to XCEx.<br>For multichannel selection when XMCM = 10b (all channels enabled but masked unless selected):<br>Mask the channel that is mapped to XCEx.<br>For multichannel selection when XMCM = 11b (all channels masked unless selected):<br>Mask the channel that is mapped to XCEx. Even if the channel is enabled by the corresponding receive channel enable bit, this channel's data cannot appear on the DX pin.<br>1h (R/W) = For multichannel selection when XMCM = 01b (all channels disabled unless selected):<br>Enable and unmask the channel that is mapped to XCEx.<br>For multichannel selection when XMCM = 10b (all channels enabled but masked unless selected):<br>Unmask the channel that is mapped to XCEx.<br>For multichannel selection when XMCM = 11b (all channels masked unless selected):<br>Unmask the channel that is mapped to XCEx. If the channel is also enabled by the corresponding receive channel enable bit, full transmission can occur. |



Copyright © 2024 Texas Instruments Incorporated

### 21.16.2.24 RCERE Register (Offset = 17h) [Reset = 0000h]

RCERE is shown in Figure 21-90 and described in Table 21-102.

Return to the Summary Table.

RCERE contains the receive channel enable registers for the E partition. This register is only used when the receiver is configured to allow individual disabling/enabling and masking/unmasking of the channels (e.g. RMCM is nonzero).

**Figure 21-90. RCERE Register**

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

**Table 21-102. RCERE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RCEE | R/W | 0h | Receive channel enable bit.<br>For receive multichannel selection mode (RMCM = 1):<br>Reset type: SYSRSn<br>0h (R/W) = Disable the channel that is mapped to RCEx.<br>1h (R/W) = Enable the channel that is mapped to RCEx. |



Copyright © 2024 Texas Instruments Incorporated

### 21.16.2.25 RCERF Register (Offset = 18h) [Reset = 0000h]

RCERF is shown in Figure 21-91 and described in Table 21-103.

Return to the Summary Table.

RCERF contains the receive channel enable registers for the F partition. This register is only used when the receiver is configured to allow individual disabling/enabling and masking/unmasking of the channels (e.g. RMCM is nonzero).

**Figure 21-91. RCERF Register**

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

**Table 21-103. RCERF Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RCEF | R/W | 0h | Receive channel enable bit.<br>For receive multichannel selection mode (RMCM = 1):<br>Reset type: SYSRSn<br>0h (R/W) = Disable the channel that is mapped to RCEx.<br>1h (R/W) = Enable the channel that is mapped to RCEx. |



Copyright © 2024 Texas Instruments Incorporated

### 21.16.2.26 XCERE Register (Offset = 19h) [Reset = 0000h]

XCERE is shown in Figure 21-92 and described in Table 21-104.

Return to the Summary Table.

XCERE contains the transmit channel enable registers for the E partition. This register is only used when the transmitter is configured to allow individual disabling/enabling and masking/unmasking of the channels (e.g. XMCM is nonzero).

**Figure 21-92. XCERE Register**

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

**Table 21-104. XCERE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | XCERE | R/W | 0h | Transmit channel enable bit.<br>The role of this bit depends on which transmit multichannel selection mode is selected with the XMCM bits.<br>Reset type: SYSRSn<br>0h (R/W) = For multichannel selection when XMCM = 01b (all channels disabled unless selected):<br>Disable and mask the channel that is mapped to XCEx.<br>For multichannel selection when XMCM = 10b (all channels enabled but masked unless selected):<br>Mask the channel that is mapped to XCEx.<br>For multichannel selection when XMCM = 11b (all channels masked unless selected):<br>Mask the channel that is mapped to XCEx. Even if the channel is enabled by the corresponding receive channel enable bit, this channel's data cannot appear on the DX pin.<br>1h (R/W) = For multichannel selection when XMCM = 01b (all channels disabled unless selected):<br>Enable and unmask the channel that is mapped to XCEx.<br>For multichannel selection when XMCM = 10b (all channels enabled but masked unless selected):<br>Unmask the channel that is mapped to XCEx.<br>For multichannel selection when XMCM = 11b (all channels masked unless selected):<br>Unmask the channel that is mapped to XCEx. If the channel is also enabled by the corresponding receive channel enable bit, full transmission can occur. |



Copyright © 2024 Texas Instruments Incorporated

### 21.16.2.27 XCERF Register (Offset = 1Ah) [Reset = 0000h]

XCERF is shown in Figure 21-93 and described in Table 21-105.

Return to the Summary Table.

XCERF contains the transmit channel enable registers for the F partition. This register is only used when the transmitter is configured to allow individual disabling/enabling and masking/unmasking of the channels (e.g. XMCM is nonzero).

**Figure 21-93. XCERF Register**

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

**Table 21-105. XCERF Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | XCERF | R/W | 0h | Transmit channel enable bit.<br>The role of this bit depends on which transmit multichannel selection mode is selected with the XMCM bits.<br>Reset type: SYSRSn<br>0h (R/W) = For multichannel selection when XMCM = 01b (all channels disabled unless selected):<br>Disable and mask the channel that is mapped to XCEx.<br>For multichannel selection when XMCM = 10b (all channels enabled but masked unless selected):<br>Mask the channel that is mapped to XCEx.<br>For multichannel selection when XMCM = 11b (all channels masked unless selected):<br>Mask the channel that is mapped to XCEx. Even if the channel is enabled by the corresponding receive channel enable bit, this channel's data cannot appear on the DX pin.<br>1h (R/W) = For multichannel selection when XMCM = 01b (all channels disabled unless selected):<br>Enable and unmask the channel that is mapped to XCEx.<br>For multichannel selection when XMCM = 10b (all channels enabled but masked unless selected):<br>Unmask the channel that is mapped to XCEx.<br>For multichannel selection when XMCM = 11b (all channels masked unless selected):<br>Unmask the channel that is mapped to XCEx. If the channel is also enabled by the corresponding receive channel enable bit, full transmission can occur. |



Copyright © 2024 Texas Instruments Incorporated

### 21.16.2.28 RCERG Register (Offset = 1Bh) [Reset = 0000h]

RCERG is shown in Figure 21-94 and described in Table 21-106.

Return to the Summary Table.

RCERG contains the receive channel enable registers for the G partition. This register is only used when the receiver is configured to allow individual disabling/enabling and masking/unmasking of the channels (e.g. RMCM is nonzero).

**Figure 21-94. RCERG Register**

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

**Table 21-106. RCERG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RCEG | R/W | 0h | Receive channel enable bit.<br>For receive multichannel selection mode (RMCM = 1):<br>Reset type: SYSRSn<br>0h (R/W) = Disable the channel that is mapped to RCEx.<br>1h (R/W) = Enable the channel that is mapped to RCEx. |



Copyright © 2024 Texas Instruments Incorporated

### 21.16.2.29 RCERH Register (Offset = 1Ch) [Reset = 0000h]

RCERH is shown in Figure 21-95 and described in Table 21-107.

Return to the Summary Table.

RCERH contains the receive channel enable registers for the H partition. This register is only used when the receiver is configured to allow individual disabling/enabling and masking/unmasking of the channels (e.g. RMCM is nonzero).

**Figure 21-95. RCERH Register**

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

**Table 21-107. RCERH Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RCEH | R/W | 0h | Receive channel enable bit.<br>For receive multichannel selection mode (RMCM = 1):<br>Reset type: SYSRSn<br>0h (R/W) = Disable the channel that is mapped to RCEx.<br>1h (R/W) = Enable the channel that is mapped to RCEx. |



Copyright © 2024 Texas Instruments Incorporated

### 21.16.2.30 XCERG Register (Offset = 1Dh) [Reset = 0000h]

XCERG is shown in Figure 21-96 and described in Table 21-108.

Return to the Summary Table.

XCERG contains the transmit channel enable registers for the G partition. This register is only used when the transmitter is configured to allow individual disabling/enabling and masking/unmasking of the channels (e.g. XMCM is nonzero).

**Figure 21-96. XCERG Register**

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

**Table 21-108. XCERG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | XCERG | R/W | 0h | Transmit channel enable bit.<br>The role of this bit depends on which transmit multichannel selection mode is selected with the XMCM bits.<br>Reset type: SYSRSn<br>0h (R/W) = For multichannel selection when XMCM = 01b (all channels disabled unless selected):<br>Disable and mask the channel that is mapped to XCEx.<br>For multichannel selection when XMCM = 10b (all channels enabled but masked unless selected):<br>Mask the channel that is mapped to XCEx.<br>For multichannel selection when XMCM = 11b (all channels masked unless selected):<br>Mask the channel that is mapped to XCEx. Even if the channel is enabled by the corresponding receive channel enable bit, this channel's data cannot appear on the DX pin.<br>1h (R/W) = For multichannel selection when XMCM = 01b (all channels disabled unless selected):<br>Enable and unmask the channel that is mapped to XCEx.<br>For multichannel selection when XMCM = 10b (all channels enabled but masked unless selected):<br>Unmask the channel that is mapped to XCEx.<br>For multichannel selection when XMCM = 11b (all channels masked unless selected):<br>Unmask the channel that is mapped to XCEx. If the channel is also enabled by the corresponding receive channel enable bit, full transmission can occur. |



Copyright © 2024 Texas Instruments Incorporated

### 21.16.2.31 XCERH Register (Offset = 1Eh) [Reset = 0000h]

XCERH is shown in Figure 21-97 and described in Table 21-109.

Return to the Summary Table.

XCERH contains the transmit channel enable registers for the H partition. This register is only used when the transmitter is configured to allow individual disabling/enabling and masking/unmasking of the channels (e.g. XMCM is nonzero).

**Figure 21-97. XCERH Register**

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

**Table 21-109. XCERH Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | XCERH | R/W | 0h | Transmit channel enable bit.<br>The role of this bit depends on which transmit multichannel selection mode is selected with the XMCM bits.<br>Reset type: SYSRSn<br>0h (R/W) = For multichannel selection when XMCM = 01b (all channels disabled unless selected):<br>Disable and mask the channel that is mapped to XCEx.<br>For multichannel selection when XMCM = 10b (all channels enabled but masked unless selected):<br>Mask the channel that is mapped to XCEx.<br>For multichannel selection when XMCM = 11b (all channels masked unless selected):<br>Mask the channel that is mapped to XCEx. Even if the channel is enabled by the corresponding receive channel enable bit, this channel's data cannot appear on the DX pin.<br>1h (R/W) = For multichannel selection when XMCM = 01b (all channels disabled unless selected):<br>Enable and unmask the channel that is mapped to XCEx.<br>For multichannel selection when XMCM = 10b (all channels enabled but masked unless selected):<br>Unmask the channel that is mapped to XCEx.<br>For multichannel selection when XMCM = 11b (all channels masked unless selected):<br>Unmask the channel that is mapped to XCEx. If the channel is also enabled by the corresponding receive channel enable bit, full transmission can occur. |



Copyright © 2024 Texas Instruments Incorporated

### 21.16.2.32 MFFINT Register (Offset = 23h) [Reset = 0000h]

MFFINT is shown in Figure 21-98 and described in Table 21-110.

Return to the Summary Table.

MFFINT contains the enable bits for both the transmitter and receiver interupts.

**Figure 21-98. MFFINT Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | RINT | RESERVED | XINT |
| R-0h | | | | | R/W-0h | R-0h | R/W-0h |

**Table 21-110. MFFINT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-3 | RESERVED | R | 0h | Reserved |
| 2 | RINT | R/W | 0h | Enable for Receive Interrupt<br>Reset type: SYSRSn<br>0h (R/W) = Receive interrupt on RRDY is disabled.<br>1h (R/W) = Receive interrupt on RRDY is enabled. |
| 1 | RESERVED | R | 0h | Reserved |
| 0 | XINT | R/W | 0h | Enable for Transmit Interrupt<br>Reset type: SYSRSn<br>0h (R/W) = Transmit interrupt on XRDY is disabled.<br>1h (R/W) = Transmit interrupt on XRDY is enabled. |

### *21.16.3 MCBSP Registers to Driverlib Functions*

**Table 21-111. MCBSP Registers to Driverlib Functions**

| File | Driverlib Function |
|---|---|
| **DRR2** | |
| mcbsp.h | McBSP_read32bitData |
| **DRR1** | |
| mcbsp.h | McBSP_read16bitData |
| mcbsp.h | McBSP_read32bitData |
| **DXR2** | |
| mcbsp.h | McBSP_write32bitData |
| **DXR1** | |
| mcbsp.h | McBSP_write16bitData |
| mcbsp.h | McBSP_write32bitData |
| **SPCR2** | |
| mcbsp.h | McBSP_setEmulationMode |
| mcbsp.h | McBSP_resetFrameSyncLogic |
| mcbsp.h | McBSP_enableFrameSyncLogic |
| mcbsp.h | McBSP_resetSampleRateGenerator |
| mcbsp.h | McBSP_enableSampleRateGenerator |
| mcbsp.h | McBSP_setTxInterruptSource |
| mcbsp.h | McBSP_getTxErrorStatus |
| mcbsp.h | McBSP_clearTxFrameSyncError |



Copyright © 2024 Texas Instruments Incorporated

**Table 21-111. MCBSP Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| mcbsp.h | McBSP_isTxReady |
| mcbsp.h | McBSP_resetTransmitter |
| mcbsp.h | McBSP_enableTransmitter |
| **SPCR1** | |
| mcbsp.h | McBSP_disableLoopback |
| mcbsp.h | McBSP_enableLoopback |
| mcbsp.h | McBSP_setRxSignExtension |
| mcbsp.h | McBSP_setClockStopMode |
| mcbsp.h | McBSP_disableDxPinDelay |
| mcbsp.h | McBSP_enableDxPinDelay |
| mcbsp.h | McBSP_setRxInterruptSource |
| mcbsp.h | McBSP_clearRxFrameSyncError |
| mcbsp.h | McBSP_getRxErrorStatus |
| mcbsp.h | McBSP_isRxReady |
| mcbsp.h | McBSP_resetReceiver |
| mcbsp.h | McBSP_enableReceiver |
| **RCR2** | |
| mcbsp.c | McBSP_setRxDataSize |
| mcbsp.h | McBSP_disableTwoPhaseRx |
| mcbsp.h | McBSP_enableTwoPhaseRx |
| mcbsp.h | McBSP_setRxCompandingMode |
| mcbsp.h | McBSP_disableRxFrameSyncErrorDetection |
| mcbsp.h | McBSP_enableRxFrameSyncErrorDetection |
| mcbsp.h | McBSP_setRxDataDelayBits |
| **RCR1** | |
| mcbsp.c | McBSP_setRxDataSize |
| **XCR2** | |
| mcbsp.c | McBSP_setTxDataSize |
| mcbsp.h | McBSP_disableTwoPhaseTx |
| mcbsp.h | McBSP_enableTwoPhaseTx |
| mcbsp.h | McBSP_setTxCompandingMode |
| mcbsp.h | McBSP_disableTxFrameSyncErrorDetection |
| mcbsp.h | McBSP_enableTxFrameSyncErrorDetection |
| mcbsp.h | McBSP_setTxDataDelayBits |
| **XCR1** | |
| mcbsp.c | McBSP_setTxDataSize |
| **SRGR2** | |
| mcbsp.h | McBSP_setFrameSyncPulsePeriod |
| mcbsp.h | McBSP_disableSRGSyncFSR |
| mcbsp.h | McBSP_enableSRGSyncFSR |
| mcbsp.h | McBSP_setRxSRGClockSource |
| mcbsp.h | McBSP_setTxSRGClockSource |
| mcbsp.h | McBSP_setTxInternalFrameSyncSource |
| **SRGR1** | |
| mcbsp.h | McBSP_setFrameSyncPulseWidthDivider |



Copyright © 2024 Texas Instruments Incorporated

**Table 21-111. MCBSP Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| mcbsp.h | McBSP_setSRGDataClockDivider |
| **MCR2** | |
| mcbsp.h | McBSP_setTxMultichannelPartition |
| mcbsp.h | McBSP_setTxTwoPartitionBlock |
| mcbsp.h | McBSP_getTxActiveBlock |
| mcbsp.h | McBSP_setTxChannelMode |
| **MCR1** | |
| mcbsp.h | McBSP_setRxMultichannelPartition |
| mcbsp.h | McBSP_setRxTwoPartitionBlock |
| mcbsp.h | McBSP_getRxActiveBlock |
| mcbsp.h | McBSP_setRxChannelMode |
| **RCERA** | |
| mcbsp.c | McBSP_disableRxChannel |
| mcbsp.c | McBSP_enableRxChannel |
| **RCERB** | |
| - | See RCERA |
| **XCERA** | |
| mcbsp.c | McBSP_disableTxChannel |
| mcbsp.c | McBSP_enableTxChannel |
| **XCERB** | |
| - | See XCERA |
| **PCR** | |
| mcbsp.h | McBSP_setRxSRGClockSource |
| mcbsp.h | McBSP_setTxSRGClockSource |
| mcbsp.h | McBSP_setTxFrameSyncSource |
| mcbsp.h | McBSP_setRxFrameSyncSource |
| mcbsp.h | McBSP_setTxClockSource |
| mcbsp.h | McBSP_setRxClockSource |
| mcbsp.h | McBSP_setTxFrameSyncPolarity |
| mcbsp.h | McBSP_setRxFrameSyncPolarity |
| mcbsp.h | McBSP_setTxClockPolarity |
| mcbsp.h | McBSP_setRxClockPolarity |
| **RCERC** | |
| - | See RCERA |
| **RCERD** | |
| - | See RCERA |
| **XCERC** | |
| - | See XCERA |
| **XCERD** | |
| - | See XCERA |
| **RCERE** | |
| - | See RCERA |
| **RCERF** | |
| - | See RCERA |
| **XCERE** | |



Copyright © 2024 Texas Instruments Incorporated

**Table 21-111. MCBSP Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| - | See XCERA |
| **XCERF** | |
| - | See XCERA |
| **RCERG** | |
| - | See RCERA |
| **RCERH** | |
| - | See RCERA |
| **XCERG** | |
| - | See XCERA |
| **XCERH** | |
| - | See XCERA |
| **MFFINT** | |
| mcbsp.h | McBSP_enableRxInterrupt |
| mcbsp.h | McBSP_disableRxInterrupt |
| mcbsp.h | McBSP_enableTxInterrupt |
| mcbsp.h | McBSP_disableTxInterrupt |



Copyright © 2024 Texas Instruments Incorporated

Chapter 22

# 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*


**22.1 Introduction** ..... 2535
**22.2 Functional Description** ..... 2538
**22.3 Operating Modes** ..... 2540
**22.4 Multiple Clock Source** ..... 2546
**22.5 Interrupt Functionality** ..... 2547
**22.6 Parity Check Mechanism** ..... 2549
**22.7 Debug Mode** ..... 2550
**22.8 Module Initialization** ..... 2550
**22.9 Configuration of Message Objects** ..... 2551
**22.10 Message Handling** ..... 2552
**22.11 CAN Bit Timing** ..... 2558
**22.12 Message Interface Register Sets** ..... 2567
**22.13 Message RAM** ..... 2569
**22.14 Software** ..... 2574
**22.15 CAN Registers** ..... 2574





Copyright © 2024 Texas Instruments Incorporated

## 22.1 Introduction

This device uses the CAN IP known as DCAN.

### *22.1.1 DCAN Related Collateral*

**Foundational Materials**

- Automotive CAN Overview and Training (Video)
- C2000 Academy - CAN
  - Refer to the DCAN section
- CAN Physical layer (Video)
- CAN and CAN FD Overview (Video)
- CAN and CAN FD Protocol (Video)

**Getting Started Materials**

- *Programming Examples and Debug Strategies for the DCAN Module Application Report*

**Expert Materials**

- *Configurable Error Generator for Controller Area Network Application Report*

### *22.1.2 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 200MHz, the smallest bit rate possible is 7.8125kbps.

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.



Copyright © 2024 Texas Instruments Incorporated

### 22.1.3 Block Diagram

Figure 22-1 shows a block diagram of the CAN module. Following is a description of some of the main blocks.

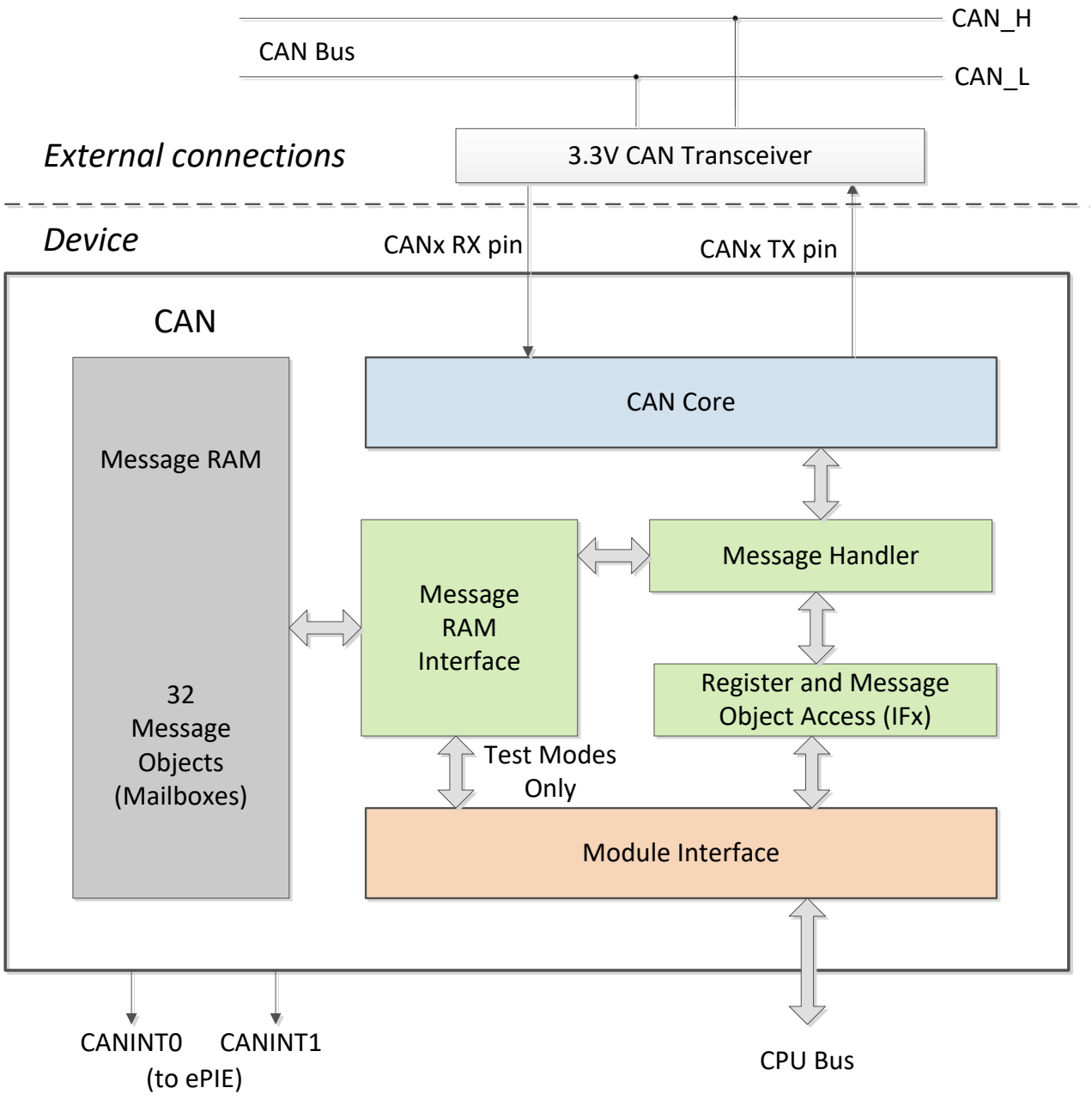



**Figure 22-1. CAN Block Diagram**



Copyright © 2024 Texas Instruments Incorporated

#### 22.1.3.1 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.

#### 22.1.3.2 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.

#### 22.1.3.3 Message RAM

The CAN message RAM enables the storage of 32 CAN messages.

#### 22.1.3.4 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 22-2. 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 22.12. 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.

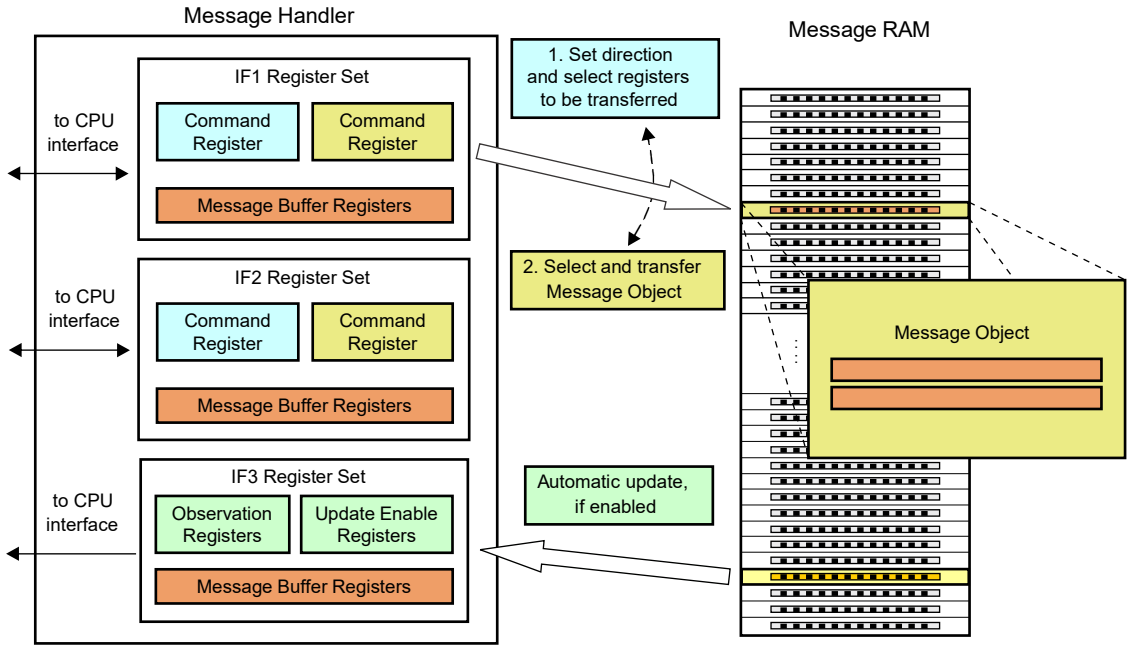



**Figure 22-2. Accessing Message Objects Through IFx Registers**



Copyright © 2024 Texas Instruments Incorporated

## 22.2 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.

### *22.2.1 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.

### *22.2.2 Address/Data Bus Bridge*

The CAN module uses a special addressing scheme to support byte accesses.This is the same addressing scheme that is used on the USB module. 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.



Copyright © 2024 Texas Instruments Incorporated

**Table 22-1. 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 | 0xBBAA9988 | __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))) | 0xBBAA9988 |
| | | | __byte((int *)0x09,0) | 0x0099 | (*((short *)(0x09))) | 0x9988 | (*((long *)(0x09))) | 0xBBAA9988 |
| | | | __byte((int *)0x0A,0) | 0x00AA | (*((short *)(0x0A))) | 0xBBAA | (*((long *)(0x0A))) | 0xBBAA9988 |
| | | | __byte((int *)0x0B,0) | 0x00BB | (*((short *)(0x0B))) | 0xBBAA | (*((long *)(0x0B))) | 0xBBAA9988 |
| | | | __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 22-2. 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 | | | |



Copyright © 2024 Texas Instruments Incorporated

## 22.3 Operating Modes

### *22.3.1 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 22.11.

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.



Copyright © 2024 Texas Instruments Incorporated

### 22.3.2 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.

#### 22.3.2.1 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 22.10.3.

#### 22.3.2.2 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.



Copyright © 2024 Texas Instruments Incorporated

### 22.3.3 Test Modes

The CAN module provides several test modes that are mainly intended for self-test purposes. Figure 22-3 aids in understanding the various test modes. Figure 22-3 must be viewed as representative of the module behavior, and not as a gate-accurate implementation of the module. Figure 22-3 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.

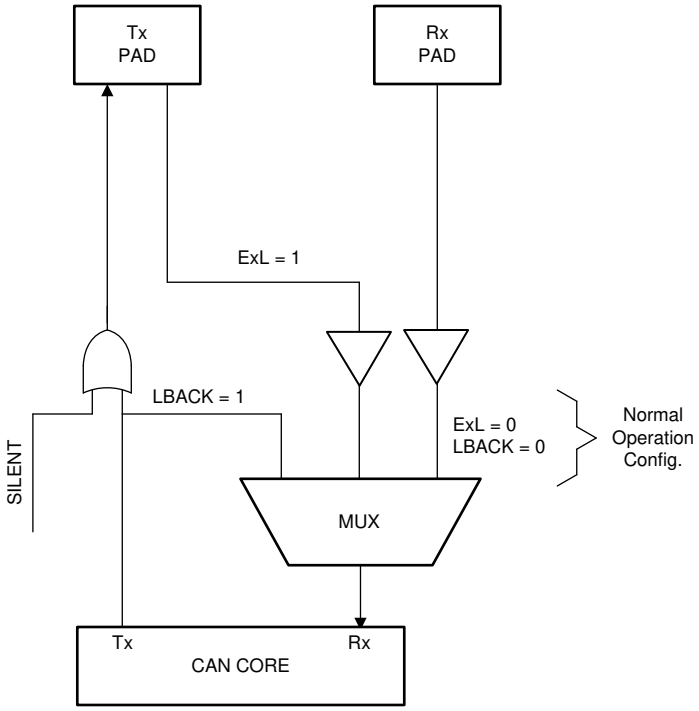



**Figure 22-3. CAN_MUX**



Copyright © 2024 Texas Instruments Incorporated

#### 22.3.3.1 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 22-4 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.

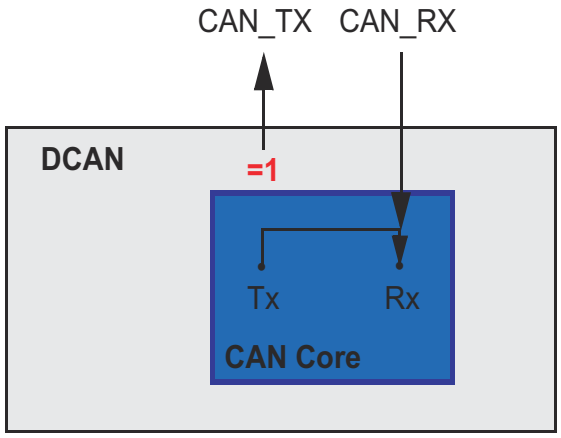



**Figure 22-4. CAN Core in Silent Mode**



Copyright © 2024 Texas Instruments Incorporated

#### 22.3.3.2 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 22-5 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 22.3.3.3.

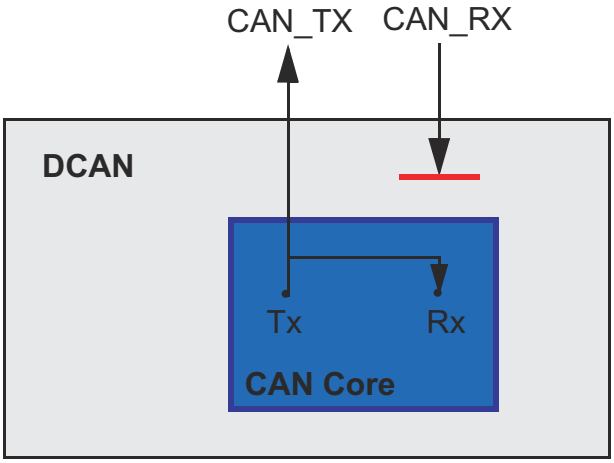



**Figure 22-5. CAN Core in Loopback Mode**



Copyright © 2024 Texas Instruments Incorporated

#### 22.3.3.3 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 22-6 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.

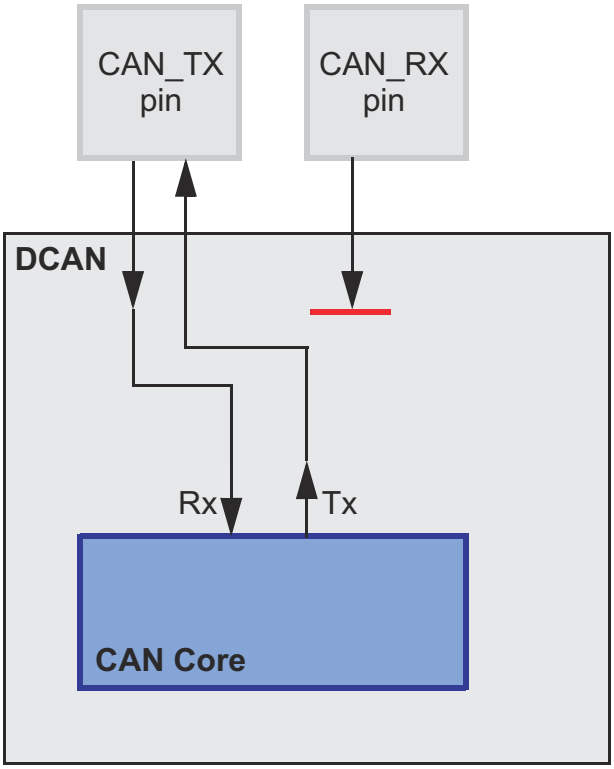



**Figure 22-6. CAN Core in External Loopback Mode**



Copyright © 2024 Texas Instruments Incorporated

#### 22.3.3.4 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 22-7 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.

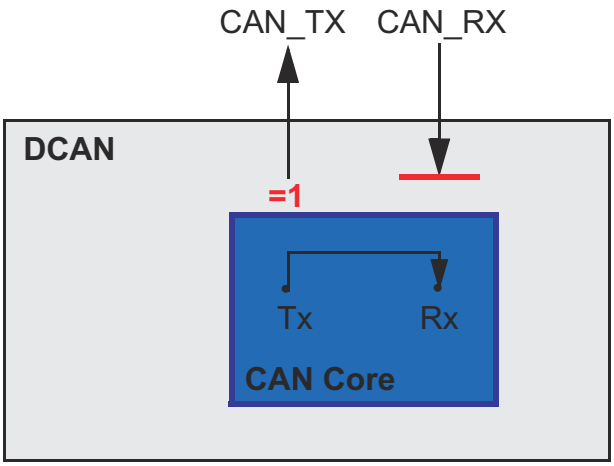



**Figure 22-7. CAN Core in Loopback Combined with Silent Mode**

## 22.4 Multiple Clock Source

Three clock domains are provided to the CAN module for generating the CAN bit timing: the external clock (X1/X2), the system clock (SYSCLK), and the GPIO_AUXCLKIN.

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.



Copyright © 2024 Texas Instruments Incorporated

## 22.5 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 CAN_INT 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.

### *22.5.1 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 22.13.1. 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.

### *22.5.2 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.

### *22.5.3 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.

### *22.5.4 Peripheral Interrupt Expansion (PIE) Module Nomenclature for DCAN Interrupts*

Table 22-3 shows the Peripheral Interrupt Expansion (PIE) module nomenclature for the interrupts.

**Table 22-3. PIE Module Nomenclature for Interrupts**

| Interrupt | CANA | CANB |
|---|---|---|
| CANINT0 | CANA_0 | CANB_0 |
| CANINT1 | CANA_1 | CANB_1 |



Copyright © 2024 Texas Instruments Incorporated

### 22.5.5 Interrupt Topologies

Interrupt topologies for CAN are illustrated in Figure 22-8 and Figure 22-9. 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.

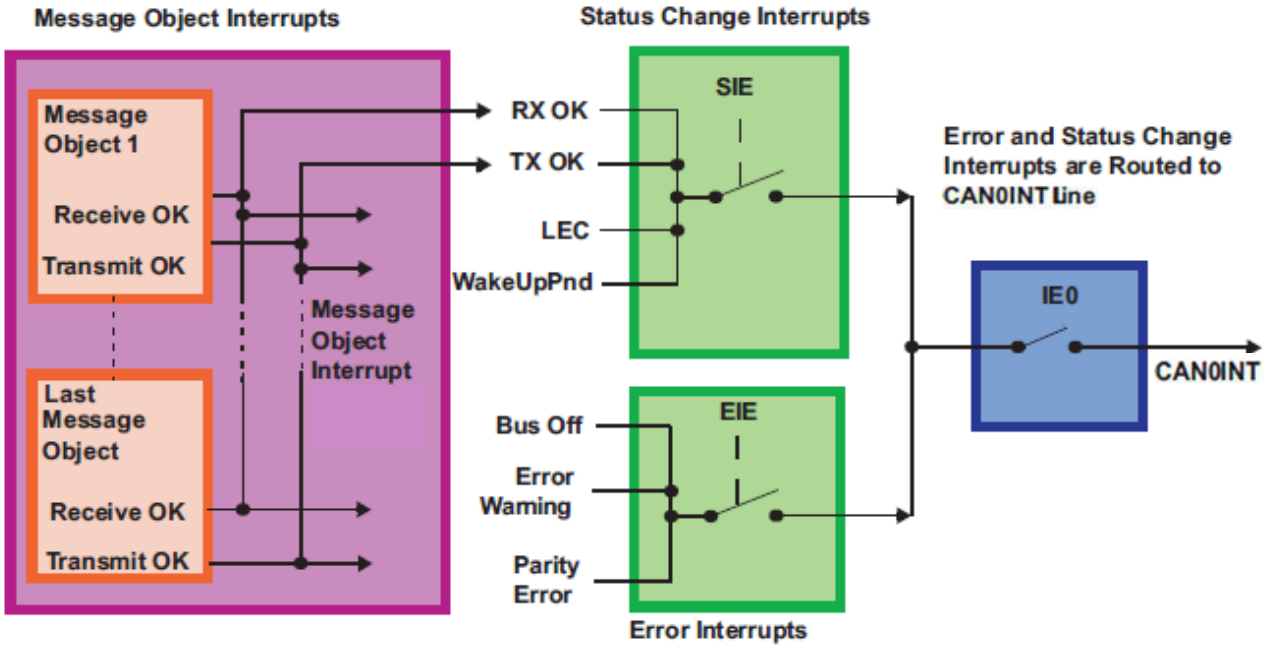



**Figure 22-8. CAN Interrupt Topology 1**

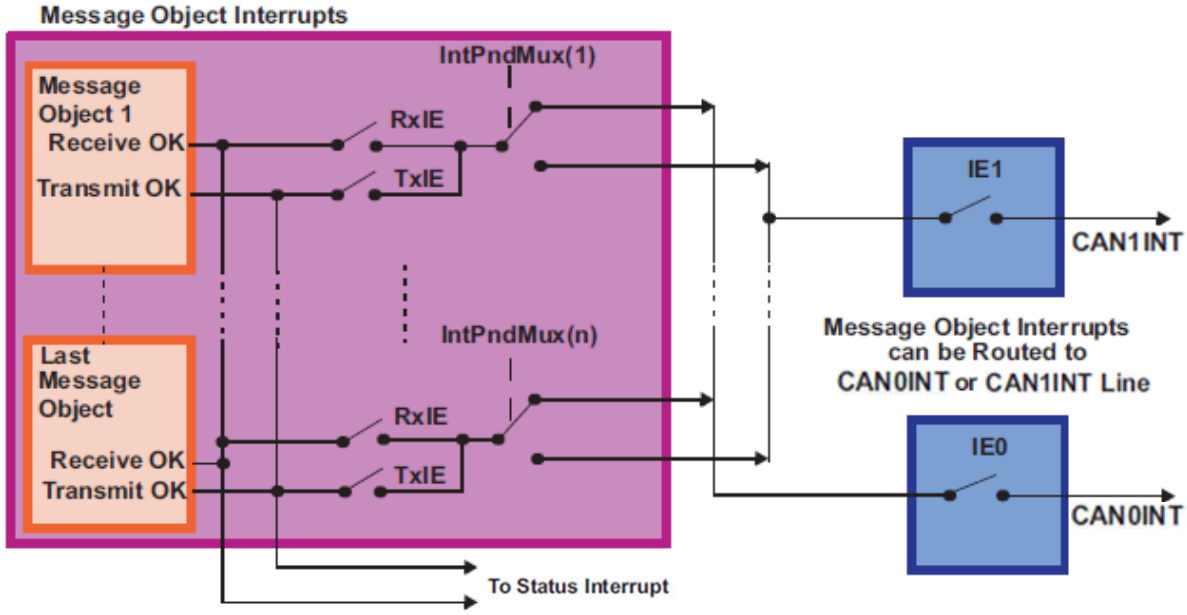



**Figure 22-9. CAN Interrupt Topology 2**



Copyright © 2024 Texas Instruments Incorporated

## 22.6 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.

### *22.6.1 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.



Copyright © 2024 Texas Instruments Incorporated

## 22.7 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 22.13.3).

**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

## 22.8 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 22.9.

For the configuration of the Bit Timing, see Section 22.11.2.

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.



Copyright © 2024 Texas Instruments Incorporated

## 22.9 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.

### *22.9.1 Configuration of a Transmit Object for Data Frames*

Figure 22-10 shows how a transmit object can be initialized.

**Figure 22-10. 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 22.10.8. Identifier masking must be disabled (UMask = 0) if no remote frames are allowed to set the TxRqst bit (RmtEn = 0).

### *22.9.2 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.

### *22.9.3 Configuration of a Single Receive Object for Data Frames*

Figure 22-11 shows how a receive object for data frames can be initialized.

**Figure 22-11. 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.



Copyright © 2024 Texas Instruments Incorporated

### 22.9.4 Configuration of a Single Receive Object for Remote Frames

Figure 22-12 shows how a receive object for remote frames can be initialized.

**Figure 22-12. 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 22.9.5.
- 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 22.10.8.
- 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.

### 22.9.5 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.

## 22.10 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.



Copyright © 2024 Texas Instruments Incorporated

### 22.10.1 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.

### 22.10.2 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.

### 22.10.3 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.



Copyright © 2024 Texas Instruments Incorporated

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.

### *22.10.4 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 22.10.3.

When NewDat is set together with TxRqst, NewDat is reset as soon as the new transmission has started.

### *22.10.5 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.



Copyright © 2024 Texas Instruments Incorporated

### 22.10.6 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.

### 22.10.7 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.

### 22.10.8 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

### 22.10.9 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.



Copyright © 2024 Texas Instruments Incorporated

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.

### 22.10.10 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.

### 22.10.11 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.

### 22.10.12 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 22-13.

**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.



Copyright © 2024 Texas Instruments Incorporated

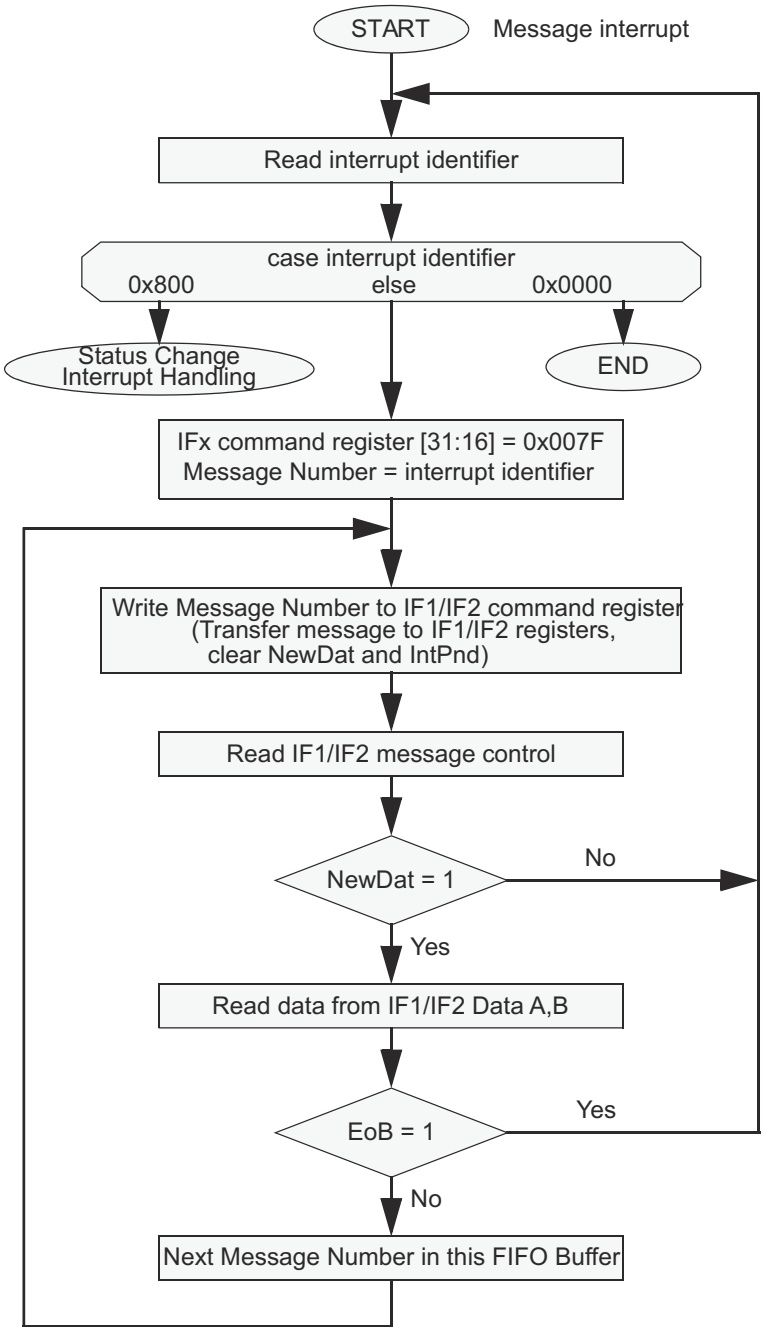



**Figure 22-13. CPU Handling of a FIFO Buffer (Interrupt Driven)**



Copyright © 2024 Texas Instruments Incorporated

## 22.11 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.

### *22.11.1 Bit Time and Bit Rate*

According to the CAN specification, the Bit time is divided into four segments (see Figure 22-14):

- Synchronization Segment (Sync_Seg)
- Propagation Time Segment (Prop_Seg)
- Phase Buffer Segment 1 (Phase_Seg1)
- Phase Buffer Segment 2 (Phase_Seg2)

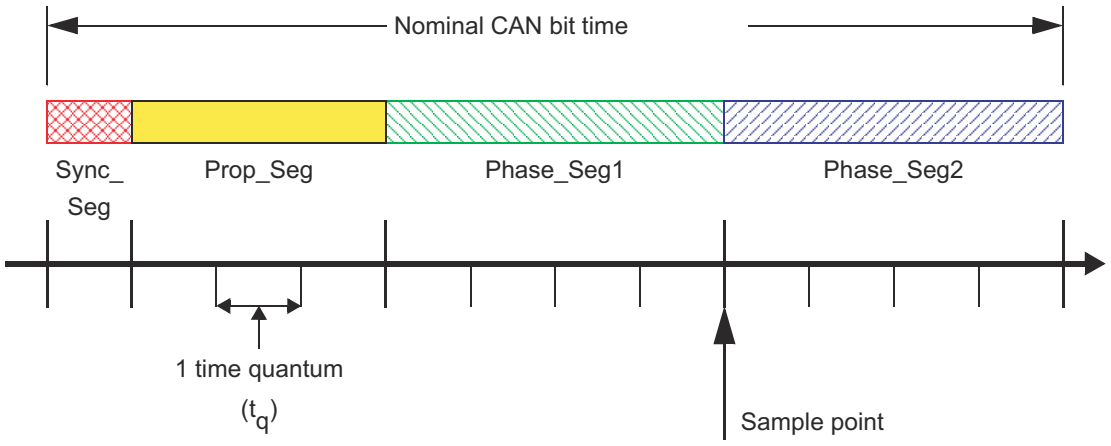



**Figure 22-14. Bit Timing**

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:

$t_q$ = Baud Rate Prescaler/CAN_CLK

Apart from the fixed length of the synchronization segment, these numbers are programmable. Table 22-4 describes the minimum programmable ranges required by the CAN protocol.

A given bit rate can be met by different bit time configurations.



Copyright © 2024 Texas Instruments Incorporated

**Table 22-4. 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.

#### 22.11.1.1 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.

#### 22.11.1.2 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 22-15 shows the phase shift and propagation times between two CAN nodes.

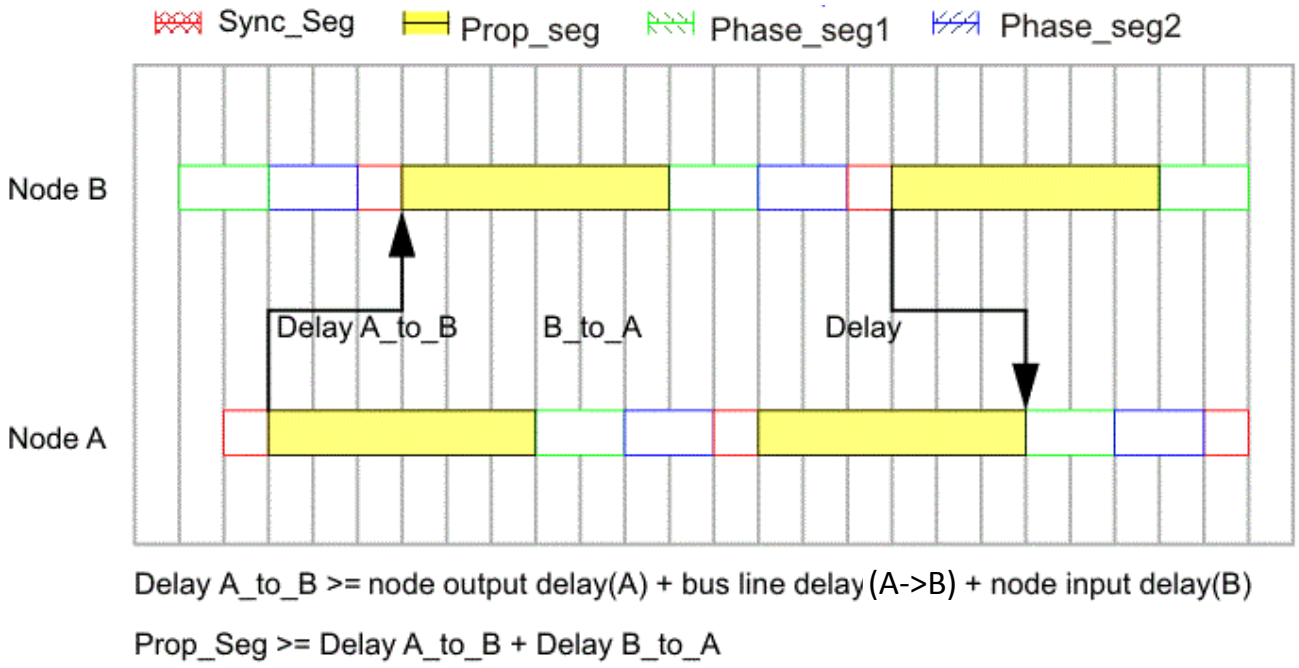



**Figure 22-15. Propagation Time Segment**



Copyright © 2024 Texas Instruments Incorporated

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.

#### 22.11.1.3 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.



Copyright © 2024 Texas Instruments Incorporated

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 22-16 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.

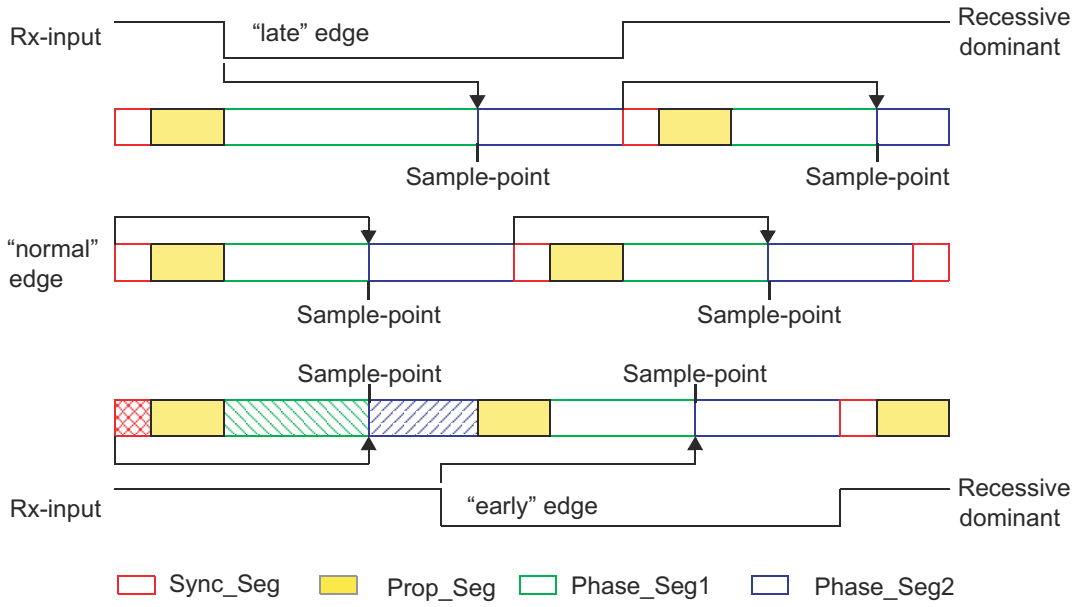



**Figure 22-16. Synchronization on Late and Early Edges**



Copyright © 2024 Texas Instruments Incorporated

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 22-17 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.

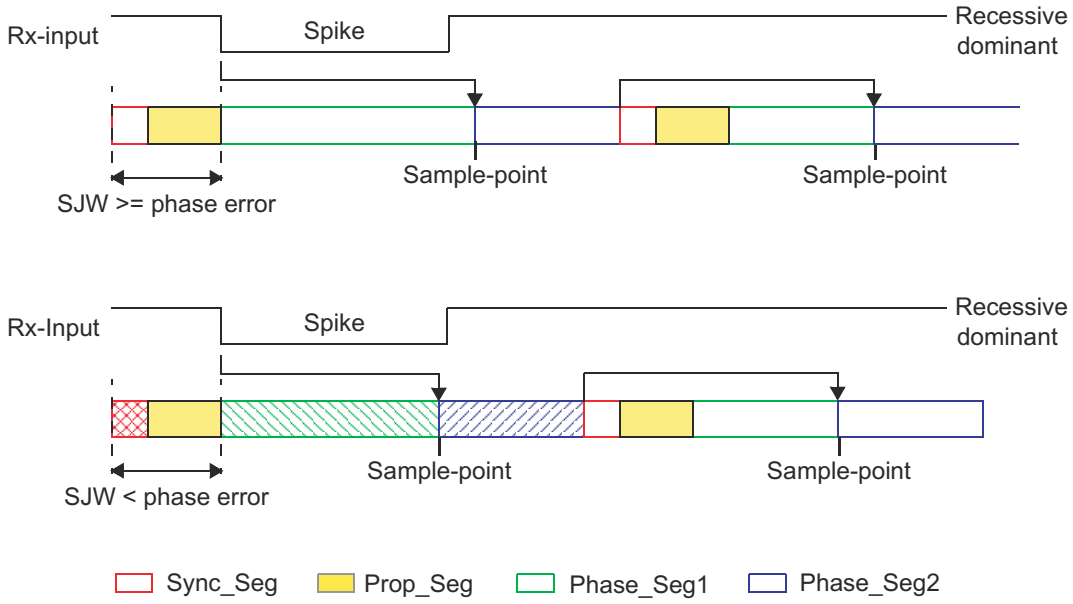



**Figure 22-17. Filtering of Short Dominant Spikes**



Copyright © 2024 Texas Instruments Incorporated

#### 22.11.1.4 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 = 8µs) with a bus length of 40 meters.

### *22.11.2 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 22-18).

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.



Copyright © 2024 Texas Instruments Incorporated

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.

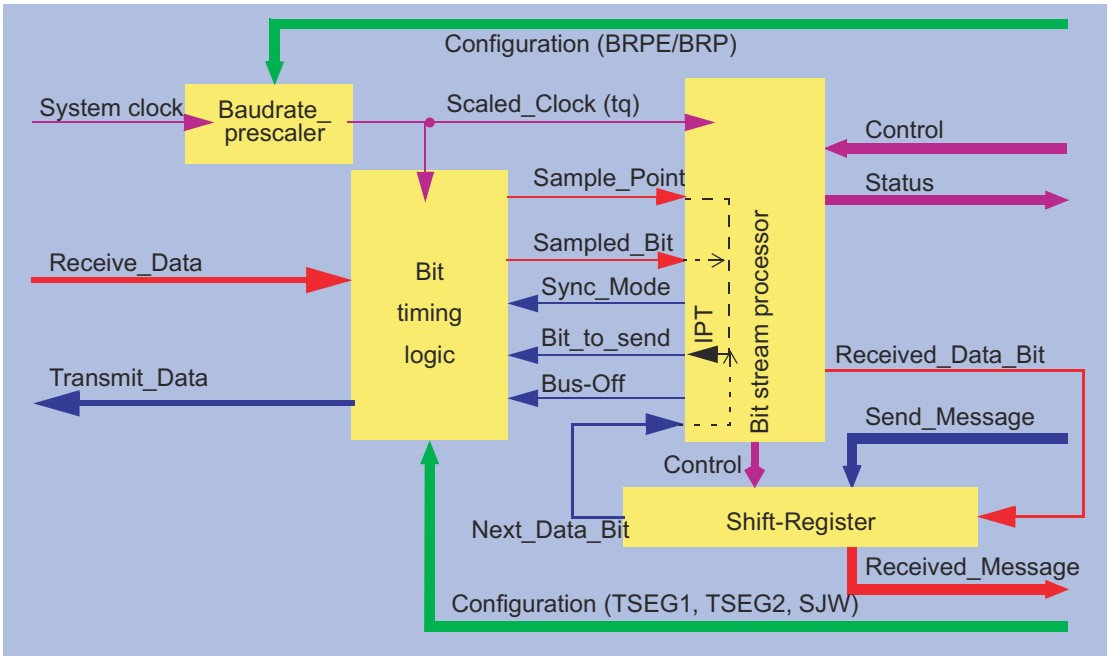



**Figure 22-18. Structure of the CAN Core's CAN Protocol Controller**



Copyright © 2024 Texas Instruments Incorporated

#### 22.11.2.1 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 $t_q$ = (Baud Rate Prescaler) / 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 (bit time - Prop_Seg - 1) $t_q$ for the two Phase Buffer Segments. If the number of remaining $t_q$ is even, the Phase Buffer Segments have the same length, Phase_Seg2 = Phase_Seg1; else, Phase_Seg2 = Phase_Seg1 + 1.

The minimum nominal length of Phase_Seg2 has to be regarded as well. Phase_Seg2 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 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 22.11.1.4.

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:

(Phase_Seg2-1)&(Phase_Seg1+Prop_Seg-1)&(SynchronizationJumpWidth-1)&(Prescaler-1)



Copyright © 2024 Texas Instruments Incorporated

#### 22.11.2.2 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*delays = 7 • $t_q$ |
| $t_{SJW}$ | 100ns | = | 1 • $t_q$ |
| $t_{TSeg1}$ | 800ns | = | $t_{Prop}$ + $t_{SJW}$ |
| $t_{TSeg2}$ | 100ns | = | Information Processing Time + 1 • $t_q$ |
| $t_{Sync-Seg}$ | 100ns | = | 1 • $t_q$ |
| bit time | 1000ns | = | $t_{Sync-Seg}$ + $t_{TSeg1}$ + $t_{TSeg2}$ |
| tolerance for CAN_CLK | 0.35% | = | $\frac{\min(Tseg1, Tseg2)}{2((13 \times bit\ time) - Tseg2)}$ $= \frac{0.1\,\mu s}{2((13 \times 1\,\mu s) - 0.1\,\mu s)}$ |

In this example, the concatenated bit time parameters are $(1\text{-}1)_3$&$(8\text{-}1)_4$&$(1\text{-}1)_2$&$(1\text{-}1)_6$, so the Bit Timing Register is programmed to = 0x0000 0700.

#### 22.11.2.3 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.

| | | | |
|---|---|---|---|
| $t_q$ | 1µs | = | 2 • $t_{CAN_CLK}$ |
| delay of bus driver | 200ns | = | |
| delay of receiver circuit | 80ns | = | |
| delay of bus line (40m) | 220ns | = | |
| $t_{Prop}$ | 1µs | = | 1 • $t_q$ |
| $t_{SJW}$ | 4µs | = | 4 • $t_q$ |
| $t_{TSeg1}$ | 5µs | = | $t_{Prop}$ + $t_{SJW}$ |
| $t_{TSeg2}$ | 4µs | = | Information Processing Time + 4 • $t_q$ |
| $t_{Sync-Seg}$ | 1µs | = | 1 • $t_q$ |
| bit time | 10µs | = | $t_{Sync-Seg}$ + $t_{TSeg1}$ + $t_{TSeg2}$ |
| tolerance for CAN_CLK | 1.58% | = | $\frac{\min(Tseg1, Tseg2)}{2((13 \times bit\ time) - Tseg2)}$ $= \frac{4\,\mu s}{2((13 \times 10\,\mu s) - 4\,\mu s)}$ |

In this example, the concatenated bit time parameters are $(4\text{-}1)_3$&$(5\text{-}1)_4$&$(4\text{-}1)_2$&$(2\text{-}1)_6$, so the Bit Timing register is programmed to = 0x0000 34C1.



Copyright © 2024 Texas Instruments Incorporated

## 22.12 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-modifywrite 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 22.13.1) 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.

### *22.12.1 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 22-19).



Copyright © 2024 Texas Instruments Incorporated

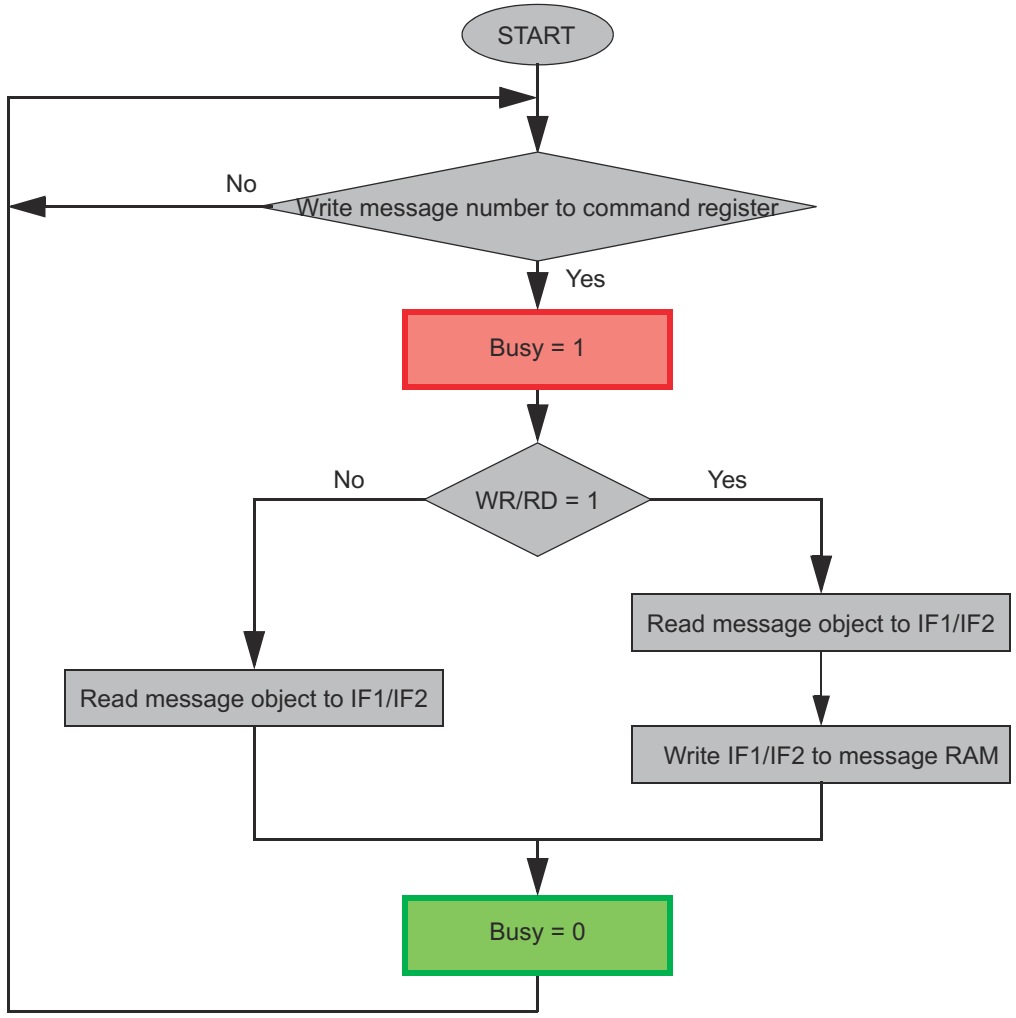



**Figure 22-19. Data Transfer Between IF1 / IF2 Registers and Message RAM**

### *22.12.2 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.

If DCAN internal IF3 update is complete, an IF3 interrupt can also be generated.

**Note**

The IF3 register set cannot be used for transferring data into message objects.



Copyright © 2024 Texas Instruments Incorporated

## 22.13 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.

### *22.13.1 Structure of Message Objects*

Figure 22-20 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 22-20. Structure of a Message Object**

| Message Object | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UMask | Msk[28:0] | MXtd | MDir | EoB | unused | NewDat | MsgLst | RxIE | TxIE | IntPnd | RmtEn | TxRqst |
| MsgVal | ID[28:0] | Xtd | Dir | DLC[3:0] | Data 0 | Data 1 | Data 2 | Data 3 | Data 4 | Data 5 | Data 6 | Data 7 |

**Table 22-5. 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.<br>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. |
| ID[28:0] | | Message Identifier |
| | ID[28:0] | 29-bit ("extended") identifier bits |
| | ID[28:18] | 11-bit ("standard") identifier bits |



Copyright © 2024 Texas Instruments Incorporated

**Table 22-5. Message Object Field Descriptions (continued)**

| Name | Value | Description |
|---|---|---|
| Msk[28:0] | | Identifier Mask |
| | 0 | 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 | | 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. |
| MXtd | | Mask Extended Identifier |
| | 0 | The extended identifier bit (IDE) has no effect on the acceptance filtering. |
| | 1 | The extended identifier bit (IDE) is used for acceptance filtering. |
| | | Note: When 11-bit ("standard") Identifiers are used for a message object, the identifiers of received data frames are written into bits ID[28:18]. For acceptance filtering, only these bits together with mask bits Msk[28:18] are considered. |
| Dir | | 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, 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 | | 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. |
| EOB | | End of Block |
| | 0 | 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 | | New Data |
| | 0 | 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 | | Message Lost (only valid for Message Objects with direction = receive) |
| | 0 | 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 | | Receive Interrupt Enable |
| | 0 | IntPnd is not triggered after the successful reception of a frame. |
| | 1 | IntPnd is triggered after the successful reception of a frame. |
| TxIE | | Transmit Interrupt Enable |
| | 0 | IntPnd is not triggered after the successful transmission of a frame. |
| | 1 | IntPnd is triggered after the successful transmission of a frame. |
| IntPnd | | 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 point to this message object, if there is no other interrupt source with higher priority. |



Copyright © 2024 Texas Instruments Incorporated

**Table 22-5. Message Object Field Descriptions (continued)**

| Name | Value | Description |
|---|---|---|
| RmtEn | | Remote Enable |
| | 0 | At the reception of a remote frame, TxRqst is not changed. |
| | 1 | At the reception of a remote frame, TxRqst is set. |
| | | Note: See Section 22.10.8 for details on the setup of RmtEn and UMask for remote frames. |
| TxRqst | | 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. |
| DLC[3:0] | | 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 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. |



Copyright © 2024 Texas Instruments Incorporated

### 22.13.2 Addressing Message Objects in RAM

The starting location of a particular message object in RAM is:

Message RAM base address + (message object number) * 0x20

This means that Message Object 1 starts at offset 0x0020; Message Object 2 starts at offset 0x0040, and so on.

**Note**

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 22-6. Message RAM Addressing in Debug Mode**

| Message Object Number | Offset From Base Address | Word Number | Debug Mode(1) |
|---|---|---|---|
| last implemented (here:32) | 0x0000 | 1 | Parity |
| | 0x0004 | 2 | MXtd,MDir,Mask |
| | 0x0008 | 3 | Xtd,Dir,ID |
| | 0x000C | 4 | Ctrl |
| | 0x0010 | 5 | Data Bytes 3-0 |
| | 0x0014 | 6 | Data Bytes 7-4 |
| 1 | 0x0020 | 1 | Parity |
| | 0x0024 | 2 | MXtd,MDir,Mask |
| | 0x0028 | 3 | Xtd,Dir,ID |
| | 0x002C | 4 | Ctrl |
| | 0x0030 | 5 | Data Bytes 3-0 |
| | 0x0034 | 6 | Data Bytes 7-4 |
| 2 | 0x0040 | 1 | Parity |
| | 0x0044 | 2 | MXtd,MDir,Mask |
| | 0x0048 | 3 | Xtd,Dir,ID |
| | 0x004C | 4 | Ctrl |
| | 0x0050 | 5 | Data Bytes 3-0 |
| | 0x0054 | 6 | Data Bytes 7-4 |
| … | … | … | … |
| 31 | 0x03E0 | 1 | Parity |
| | 0x03E4 | 2 | MXtd,MDir,Mask |
| | 0x03E8 | 3 | Xtd,Dir,ID |
| | 0x03EC | 4 | Ctrl |
| | 0x03F0 | 5 | Data Bytes 3-0 |
| | 0x03F4 | 6 | Data Bytes 7-4 |

(1) See Section 22.13.3.



Copyright © 2024 Texas Instruments Incorporated

### 22.13.3 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 22-21. 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 |



Copyright © 2024 Texas Instruments Incorporated

## 22.14 Software

### 22.14.1 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 C2000Ware Examples.

## 22.15 CAN Registers

This section describes the Controller Area Network registers.

### 22.15.1 CAN Base Addresses

**Table 22-7. CAN Base Address Table**

| Device Registers | Register Name | Start Address | End Address |
|---|---|---|---|
| CanaRegs | CAN_REGS | 0x0004_8000 | 0x0004_87FF |
| CanbRegs | CAN_REGS | 0x0004_A000 | 0x0004_A7FF |



Copyright © 2024 Texas Instruments Incorporated

### 22.15.2 CAN_REGS Registers

Table 22-8 lists the memory-mapped registers for the CAN_REGS registers. All register offset addresses not listed in Table 22-8 should be considered as reserved locations and the register contents should not be modified.

**Table 22-8. CAN_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | CAN_CTL | CAN Control Register | | Go |
| 4h | CAN_ES | Error and Status Register | | Go |
| 8h | CAN_ERRC | Error Counter Register | | Go |
| Ch | CAN_BTR | Bit Timing Register | | Go |
| 10h | CAN_INT | Interrupt Register | | Go |
| 14h | CAN_TEST | Test Register | | Go |
| 1Ch | CAN_PERR | CAN Parity Error Code Register | | Go |
| 40h | CAN_RAM_INIT | CAN RAM Initialization Register | | Go |
| 50h | CAN_GLB_INT_EN | CAN Global Interrupt Enable Register | | Go |
| 54h | CAN_GLB_INT_FLG | CAN Global Interrupt Flag Register | | Go |
| 58h | CAN_GLB_INT_CLR | CAN Global Interrupt Clear Register | | Go |
| 80h | CAN_ABOTR | Auto-Bus-On Time Register | | Go |
| 84h | CAN_TXRQ_X | CAN Transmission Request Register | | Go |
| 88h | CAN_TXRQ_21 | CAN Transmission Request 2_1 Register | | Go |
| 98h | CAN_NDAT_X | CAN New Data Register | | Go |
| 9Ch | CAN_NDAT_21 | CAN New Data 2_1 Register | | Go |
| ACh | CAN_IPEN_X | CAN Interrupt Pending Register | | Go |
| B0h | CAN_IPEN_21 | CAN Interrupt Pending 2_1 Register | | Go |
| C0h | CAN_MVAL_X | CAN Message Valid Register | | Go |
| C4h | CAN_MVAL_21 | CAN Message Valid 2_1 Register | | Go |
| D8h | CAN_IP_MUX21 | CAN Interrupt Multiplexer 2_1 Register | | Go |
| 100h | CAN_IF1CMD | IF1 Command Register | | Go |
| 104h | CAN_IF1MSK | IF1 Mask Register | | Go |
| 108h | CAN_IF1ARB | IF1 Arbitration Register | | Go |
| 10Ch | CAN_IF1MCTL | IF1 Message Control Register | | Go |
| 110h | CAN_IF1DATA | IF1 Data A Register | | Go |
| 114h | CAN_IF1DATB | IF1 Data B Register | | Go |
| 120h | CAN_IF2CMD | IF2 Command Register | | Go |
| 124h | CAN_IF2MSK | IF2 Mask Register | | Go |
| 128h | CAN_IF2ARB | IF2 Arbitration Register | | Go |
| 12Ch | CAN_IF2MCTL | IF2 Message Control Register | | Go |
| 130h | CAN_IF2DATA | IF2 Data A Register | | Go |
| 134h | CAN_IF2DATB | IF2 Data B Register | | Go |
| 140h | CAN_IF3OBS | IF3 Observation Register | | Go |
| 144h | CAN_IF3MSK | IF3 Mask Register | | Go |
| 148h | CAN_IF3ARB | IF3 Arbitration Register | | Go |
| 14Ch | CAN_IF3MCTL | IF3 Message Control Register | | Go |
| 150h | CAN_IF3DATA | IF3 Data A Register | | Go |
| 154h | CAN_IF3DATB | IF3 Data B Register | | Go |
| 160h | CAN_IF3UPD | IF3 Update Enable Register | | Go |



Copyright © 2024 Texas Instruments Incorporated

Complex bit access types are encoded to fit into small table cells. Table 22-9 shows the codes that are used for access types in this section.

**Table 22-9. CAN_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| R-0 | R<br>-0 | Read<br>Returns 0s |
| Write Type | | |
| W | W | Write |
| W1C | W<br>1C | Write<br>1 to clear |
| W1S | W<br>1S | Write<br>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. |



Copyright © 2024 Texas Instruments Incorporated

### 22.15.2.1 CAN_CTL Register (Offset = 0h) [Reset = 00001401h]

CAN_CTL is shown in Figure 22-22 and described in Table 22-10.

Return to the Summary Table.

This register is used for configuring the CAN module in terms of interrupts, parity, debug-mode behavior etc.

**Figure 22-22. 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 22-10. 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<br>0 CANINT1 is disabled.<br>1 CANINT1 is enabled. Interrupts will assert CANINT1 line to 1 line remains active until pending interrupts are processed.<br>Reset type: SYSRSn |
| 16 | INITDBG | R | 0h | Debug Mode Status Bit: This bit indicates the internal init state for a debug access<br>0 Not in debug mode, or debug mode requested but not entered.<br>1 Debug mode requested and internally entered<br>the CAN module is ready for debug accesses.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 22-10. 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.<br>0 Normal Operation.<br>1 Module is forced to reset state. This bit will get cleared automatically one clock cycle after execution of software reset.<br>Note: To execute software reset, the following procedure is necessary:<br>1. Set INIT bit to shut down CAN communication.<br>2. Set SWR bit.<br>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.<br>Reset type: SYSRSn |
| 14 | RESERVED | R | 0h | Reserved |
| 13-10 | PMD | R/W | 5h | Parity on/off<br>0101 Parity function disabled<br>Any other value - Parity function enabled<br>Reset type: SYSRSn |
| 9 | ABO | R/W | 0h | Auto-Bus-On Enable<br>0 The Auto-Bus-On feature is disabled<br>1 The Auto-Bus-On feature is enabled<br>Reset type: SYSRSn |
| 8 | IDS | R/W | 0h | Interruption Debug Support Enable<br>0 When Debug mode is requested, the CAN module will wait for a started transmission or reception to be completed before entering Debug mode<br>1 When Debug mode is requested, the CAN module will interrupt any transmission or reception, and enter Debug mode immediately.<br>Reset type: SYSRSn |
| 7 | Test | R/W | 0h | Test Mode Enable<br>0 Disable Test Mode (Normal operation)<br>1 Enable Test Mode<br>Reset type: SYSRSn |
| 6 | CCE | R/W | 0h | Configuration Change Enable<br>0 The CPU has no write access to the configuration registers.<br>1 The CPU has write access to the configuration registers (when Init bit is set).<br>Reset type: SYSRSn |
| 5 | DAR | R/W | 0h | Disable Automatic Retransmission<br>0 Automatic Retransmission of 'not successful' messages enabled.<br>1 Automatic Retransmission disabled.<br>Reset type: SYSRSn |
| 4 | RESERVED | R | 0h | Reserved |
| 3 | EIE | R/W | 0h | Error Interrupt Enable<br>0 Disabled - PER, BOff and EWarn bits cannot generate an interrupt.<br>1 Enabled - PER, BOff and EWarn bits can generate an interrupt at CANINT0 line and affect the Interrupt Register.<br>Reset type: SYSRSn |
| 2 | SIE | R/W | 0h | Status Change Interrupt Enable<br>0 Disabled - RxOk, TxOk and LEC bits cannot generate an interrupt.<br>1 Enabled - RxOk, TxOk and LEC can generate an interrupt on the CANINT0 line<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 22-10. CAN_CTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1 | IE0 | R/W | 0h | Interrupt line 0 Enable<br>0 CANINT0 is disabled.<br>1 CANINT0 is enabled. Interrupts will assert CANINT0 line to 1 line remains active until pending interrupts are processed.<br>Reset type: SYSRSn |
| 0 | Init | R/W | 1h | Initialization Mode<br>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.<br>0 CAN module processes messages normally<br>1 CAN module ignores bus activity<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 22.15.2.2 CAN_ES Register (Offset = 4h) [Reset = 00000007h]

CAN_ES is shown in Figure 22-23 and described in Table 22-11.

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 22-23. 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 22-11. 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.<br>0 No parity error has been detected since last read access.<br>1 The parity check mechanism has detected a parity error in the Message RAM.<br>Reset type: SYSRSn |
| 7 | BOff | R | 0h | Bus-off Status Bit:<br>0 The CAN module is not in Bus-Off state.<br>1 The CAN module is in Bus-Off state.<br>Reset type: SYSRSn |
| 6 | EWarn | R | 0h | Warning State Bit:<br>0 Both error counters are below the error warning limit of 96.<br>1 At least one of the error counters has reached the error warning limit of 96.<br>Reset type: SYSRSn |
| 5 | EPass | R | 0h | Error Passive State<br>0 On CAN Bus error, the CAN could send active error frames.<br>1 The CAN Core is in the error passive state as defined in the CAN Specification.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 22-11. CAN_ES Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 4 | RxOk | R | 0h | Reception status Bit: This bit indicates the status of reception. The bit will be reset after the CPU reads the register.<br>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.<br>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.<br>Reset type: SYSRSn |
| 3 | TxOk | R | 0h | Transmission status Bit: This bit indicates the status of transmission. The bit will be reset after the CPU reads the register.<br>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.<br>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.<br>Reset type: SYSRSn |
| 2-0 | LEC | R | 7h | Last Error Code<br>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.<br>0 No Error<br>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.<br>2 Form Error: A fixed format part of a received frame has the wrong format.<br>3 Ack Error: The message this CAN Core transmitted was not acknowledged by another node.<br>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.<br>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).<br>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).<br>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'.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 22.15.2.3 CAN_ERRC Register (Offset = 8h) [Reset = 00000000h]

CAN_ERRC is shown in Figure 22-24 and described in Table 22-12.

Return to the Summary Table.

This register reflects the value of the Transmit and Receive error counters

**Figure 22-24. 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 22-12. CAN_ERRC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15 | RP | R | 0h | Receive Error Passive<br>0 The Receive Error Counter is below the error passive level.<br>1 The Receive Error Counter has reached the error passive level as defined in the CAN Specification.<br>Reset type: SYSRSn |
| 14-8 | REC | R | 0h | Receive Error Counter<br>Actual state of the Receive Error Counter (values from 0 to 127).<br>Reset type: SYSRSn |
| 7-0 | TEC | R | 0h | Transmit Error Counter<br>Actual state of the Transmit Error Counter. (values from 0 to 255).<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 22.15.2.4 CAN_BTR Register (Offset = Ch) [Reset = 00002301h]

CAN_BTR is shown in Figure 22-25 and described in Table 22-13.

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 to1024 CAN_CLK periods.

**Figure 22-25. 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 22-13. 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<br>Valid programmed values are 0 to 15.<br>By programming BRPE the Baud Rate Prescaler can be extended to values up to 1024.<br>Note: This bit is Write Protected by CCE bit.<br>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.<br>The actual TSeg2 value which is interpreted for the Bit Timing will be the programmed TSeg2 value + 1.<br>Note: This bit is Write Protected by CCE bit.<br>Reset type: SYSRSn |
| 11-8 | TSEG1 | R/W | 3h | Time segment before the sample point Valid programmed values are 1 to 15.<br>The actual TSeg1 value interpreted for the Bit Timing will be the programmed TSeg1 value + 1.<br>Note: This bit is Write Protected by CCE bit.<br>Reset type: SYSRSn |
| 7-6 | SJW | R/W | 0h | Synchronization Jump Width Valid programmed values are 0 to 3.<br>The actual SJW value interpreted for the Synchronization will be the programmed SJW value + 1.<br>Note: This bit is Write Protected by CCE bit.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 22-13. CAN_BTR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 5-0 | BRP | R/W | 1h | Baud Rate Prescaler-<br>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.<br>The actual BRP value interpreted for the Bit Timing will be the programmed BRP value + 1.<br>Note: This bit is Write Protected by CCE bit.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 22.15.2.5 CAN_INT Register (Offset = 10h) [Reset = 00000000h]

CAN_INT is shown in Figure 22-26 and described in Table 22-14.

Return to the Summary Table.

This register is used to identify the source of the interrupt(s).

**Figure 22-26. 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 22-14. CAN_INT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-24 | RESERVED | R | 0h | Reserved |
| 23-16 | INT1ID | R | 0h | Interrupt 1 Cause<br>0x00 No interrupt is pending.<br>0x01-0x20 Number of message object (mailbox) which caused the interrupt.<br>0x21-0xFF Unused.<br>If several interrupts are pending, the CAN Interrupt Register will point to the pending interrupt with the highest priority.<br>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.<br>Reset type: SYSRSn |
| 15-0 | INT0ID | R | 0h | Interrupt 0 Cause<br>0x0000 - No interrupt is pending.<br>0x0001 - 0x0020 - Number of message object which caused the interrupt.<br>0x0021 - 0x7FFF - Unused.<br>0x8000 - Error and Status Register value is not 0x07.<br>0x8001 - 0xFFFF - Unused.<br>If several interrupts are pending, the CAN Interrupt Register will point to the pending interrupt with the highest priority.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 22.15.2.6 CAN_TEST Register (Offset = 14h) [Reset = 00000000h]

CAN_TEST is shown in Figure 22-27 and described in Table 22-15.

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 22-27. 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 22-15. 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:<br>0 Normal Operation.<br>1 Direct access to the RAM is enabled while in Test Mode.<br>Reset type: SYSRSn |
| 8 | EXL | R/W | 0h | External Loop Back Mode:<br>0 Disabled.<br>1 Enabled.<br>Reset type: SYSRSn |
| 7 | RX | R | 0h | Monitors the actual value of the CANRX pin:<br>0 The CAN bus is dominant.<br>1 The CAN bus is recessive.<br>Reset type: SYSRSn |
| 6-5 | TX | R/W | 0h | Control of CANTX pin:<br>00 Normal operation, CANTX is controlled by the CAN Core.<br>01 Sample Point can be monitored at CANTX pin.<br>10 CANTX pin drives a dominant value.<br>11 CANTX pin drives a recessive value.<br>Reset type: SYSRSn |
| 4 | LBACK | R/W | 0h | Loop Back Mode:<br>0 Disabled.<br>1 Enabled.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 22-15. CAN_TEST Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3 | SILENT | R/W | 0h | Silent Mode:<br>0 Disabled.<br>1 Enabled.<br>Reset type: SYSRSn |
| 2-0 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 22.15.2.7 CAN_PERR Register (Offset = 1Ch) [Reset = 00000XXXh]

CAN_PERR is shown in Figure 22-28 and described in Table 22-16.

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 22-28. 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 22-16. 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).<br>Reset type: SYSRSn |
| 7-0 | MSG_NUM | R | X | 0x01-0x21 Mailbox number where parity error has been detected<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 22.15.2.8 CAN_RAM_INIT Register (Offset = 40h) [Reset = 00000005h]

CAN_RAM_INIT is shown in Figure 22-29 and described in Table 22-17.

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 22-29. 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 22-17. 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:<br>0 Read: Initialization is on-going or initialization not initiated.<br>1 Read: Initialization complete<br>Reset type: SYSRSn |
| 4 | CAN_RAM_INIT | R/W | 0h | Initiate CAN Mailbox RAM initialization:<br>0 Read: Initialization complete or initialization not initiated.<br>Write: No action<br>1 Read: Initialization is on-going<br>Write: Initiate CAN Mailbox RAM initialization. After initialization, this bit will be automatically cleared to 0.<br>Reset type: SYSRSn |
| 3 | KEY3 | R/W | 0h | See Key 0<br>Reset type: SYSRSn |
| 2 | KEY2 | R/W | 1h | See Key 0<br>Reset type: SYSRSn |
| 1 | KEY1 | R/W | 0h | See Key 0<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 22.15.2.9 CAN_GLB_INT_EN Register (Offset = 50h) [Reset = 00000000h]

CAN_GLB_INT_EN is shown in Figure 22-30 and described in Table 22-18.

Return to the Summary Table.

This register is used to enable the interrupt lines to the PIE.

**Figure 22-30. 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 22-18. 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<br>0 CANINT1 does not generate interrupt to PIE<br>1 CANINT1 generates interrupt to PIE if interrupt condition occurs<br>Reset type: SYSRSn |
| 0 | GLBINT0_EN | R/W | 0h | Global Interrupt Enable for CANINT0<br>0 CANINT0 does not generate interrupt to PIE<br>1 CANINT0 generates interrupt to PIE if interrupt condition occurs<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 22.15.2.10 CAN_GLB_INT_FLG Register (Offset = 54h) [Reset = 00000000h]

CAN_GLB_INT_FLG is shown in Figure 22-31 and described in Table 22-19.

Return to the Summary Table.

This register indicates if and when the interrupt line to the PIE is active.

**Figure 22-31. 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 22-19. 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<br>0 No interrupt generated<br>1 Interrupt is generated due to CANINT1 (refer to CAN Interrupt Status Register for the condition)<br>Reset type: SYSRSn |
| 0 | INT0_FLG | R | 0h | CANINT0 Flag<br>0 No interrupt generated<br>1 Interrupt is generated due to CANINT0 (refer to CAN Interrupt Status Register for the condition)<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 22.15.2.11 CAN_GLB_INT_CLR Register (Offset = 58h) [Reset = 00000000h]

CAN_GLB_INT_CLR is shown in Figure 22-32 and described in Table 22-20.

Return to the Summary Table.

This register is used to clear the interrupt to the PIE.

**Figure 22-32. 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 22-20. 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<br>0 No effect<br>1 Write 1 to clear the corresponding bit of the Global Interrupt Flag Register and allow the PIE to receive another interrupt from CANINT1.<br>Reset type: SYSRSn |
| 0 | INT0_FLG_CLR | W | 0h | Global Interrupt flag clear for CANINT0<br>0 No effect<br>1 Write 1 to clear the corresponding bit of the Global Interrupt Flag Register and allow the PIE to receive another interrupt from CANINT0.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 22.15.2.12 CAN_ABOTR Register (Offset = 80h) [Reset = 00000000h]

CAN_ABOTR is shown in Figure 22-33 and described in Table 22-21.

Return to the Summary Table.

This register is used to introduce a variable delay before the Bus-off recovery sequence is started.

**Figure 22-33. 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 22-21. CAN_ABOTR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | ABO_Time | R/W | 0h | Auto-Bus-On Timer<br>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.<br>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.<br>NOTE: On write access to the CAN Control register while Auto-Bus-On timer is running, the Auto-Bus-On procedure will be aborted.<br>NOTE: During Debug mode, running Auto-Bus-On timer will be paused.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 22.15.2.13 CAN_TXRQ_X Register (Offset = 84h) [Reset = 00000000h]

CAN_TXRQ_X is shown in Figure 22-34 and described in Table 22-22.

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 22-34. 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 22-22. 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:<br>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.<br>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.<br>Reset type: SYSRSn |
| 1-0 | TxRqstReg1 | R | 0h | Transmit Request Register 1 flag:<br>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.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 22.15.2.14 CAN_TXRQ_21 Register (Offset = 88h) [Reset = 00000000h]

CAN_TXRQ_21 is shown in Figure 22-35 and described in Table 22-23.

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 22-35. 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 22-23. CAN_TXRQ_21 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | TxRqst | R | 0h | Transmission Request Bits (for all message objects)<br>0 No transmission has been requested for this message object.<br>1 The transmission of this message object is requested and is not yet done.<br>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<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 22.15.2.15 CAN_NDAT_X Register (Offset = 98h) [Reset = 00000000h]

CAN_NDAT_X is shown in Figure 22-36 and described in Table 22-24.

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 22-36. 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 22-24. 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:<br>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.<br>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.<br>Reset type: SYSRSn |
| 1-0 | NewDatReg1 | R | 0h | New Data Register 1 flag:<br>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.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 22.15.2.16 CAN_NDAT_21 Register (Offset = 9Ch) [Reset = 00000000h]

CAN_NDAT_21 is shown in Figure 22-37 and described in Table 22-25.

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 22-37. 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 22-25. CAN_NDAT_21 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | NewDat | R | 0h | New Data Bits (for all message objects)<br>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.<br>1 The message handler or the CPU has written new data into the data portion of this message object.<br>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<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 22.15.2.17 CAN_IPEN_X Register (Offset = ACh) [Reset = 00000000h]

CAN_IPEN_X is shown in Figure 22-38 and described in Table 22-26.

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 22-38. 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 22-26. 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:<br>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.<br>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.<br>Reset type: SYSRSn |
| 1-0 | IntPndReg1 | R | 0h | Interrupt Pending Register 1 flag:<br>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.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 22.15.2.18 CAN_IPEN_21 Register (Offset = B0h) [Reset = 00000000h]

CAN_IPEN_21 is shown in Figure 22-39 and described in Table 22-27.

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 22-39. 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 22-27. 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.<br>0 This mailbox is not the source of an interrupt.<br>1 This mailbox is the source of an interrupt.<br>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<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 22.15.2.19 CAN_MVAL_X Register (Offset = C0h) [Reset = 00000000h]

CAN_MVAL_X is shown in Figure 22-40 and described in Table 22-28.

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 22-40. 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 22-28. 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:<br>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.<br>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.<br>Reset type: SYSRSn |
| 1-0 | MsgValReg1 | R | 0h | Message Valid Register 1 flag:<br>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.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 22.15.2.20 CAN_MVAL_21 Register (Offset = C4h) [Reset = 00000000h]

CAN_MVAL_21 is shown in Figure 22-41 and described in Table 22-29.

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 22-41. 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 22-29. CAN_MVAL_21 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | MsgValReg | R | 0h | Message Valid Bits (for all message objects)<br>0 This message object is ignored by the message handler.<br>1 This message object is configured and will be considered by the message handler.<br>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<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 22.15.2.21 CAN_IP_MUX21 Register (Offset = D8h) [Reset = 00000000h]

CAN_IP_MUX21 is shown in Figure 22-42 and described in Table 22-30.

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 22-42. 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 22-30. CAN_IP_MUX21 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | IntMux | R/W | 0h | Interrupt Mux bits:<br>0 CANINT0 line is active if corresponding IntPnd flag is one.<br>1 CANINT1 line is active if corresponding IntPnd flag is one.<br>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<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 22.15.2.22 CAN_IF1CMD Register (Offset = 100h) [Reset = 00000001h]

CAN_IF1CMD is shown in Figure 22-43 and described in Table 22-31.

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 22-43. 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 22-31. CAN_IF1CMD Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-24 | RESERVED | R | 0h | Reserved |
| 23 | DIR | R/W | 0h | Write/Read<br>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.<br>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.<br>The other bits of IF1/IF2 Command Mask Register have different functions depending on the transfer direction.<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 22-31. CAN_IF1CMD Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 22 | Mask | R/W | 0h | Access Mask Bits<br>0 Mask bits will not be changed<br>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.<br>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]).<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |
| 21 | Arb | R/W | 0h | Access Arbitration Bits<br>0 Arbitration bits will not be changed<br>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.<br>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]).<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |
| 20 | Control | R/W | 0h | Access control bits.<br>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.<br>0 Control bits will not be changed.<br>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.<br>1 (Direction = Write): The message control bits will be transferred from the IF1 register set to the<br>message object addressed by message number (Bits [7:0]).<br>Note: This bit is write protected by the Busy bit.<br>Reset type: SYSRSn |
| 19 | ClrIntPnd | R/W | 0h | Clear Interrupt Pending Bit<br>0 IntPnd bit will not be changed<br>1 (Direction = Read): Clears IntPnd bit in the message object.<br>1 (Direction = Write): This bit is ignored.<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |
| 18 | TXRQST | R/W | 0h | Access Transmission Request (TxRqst) / New Data (NewDat) Bit<br>0 (Direction = Read): NewDat bit will not be changed.<br>0 (Direction = Write): TxRqst/NewDat bit will be handled according to the Control bit.<br>1 (Direction = Read): Clears NewDat bit in the message object.<br>1 (Direction = Write): Sets TxRqst/NewDat in message object.<br>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.<br>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.<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 22-31. CAN_IF1CMD Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 17 | DATA_A | R/W | 0h | Access Data Bytes 0-3<br>0 Data Bytes 0-3 will not be changed.<br>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.<br>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]).<br>Note: The duration of the message transfer is independent of the number of bytes to be transferred.<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |
| 16 | DATA_B | R/W | 0h | Access Data Bytes 4-7<br>0 Data Bytes 4-7 will not be changed.<br>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.<br>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]).<br>Note: The duration of the message transfer is independent of the number of bytes to be transferred.<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |
| 15 | Busy | R | 0h | Busy Flag<br>0 No transfer between IF1/IF2 Register Set and Message RAM is in progress.<br>1 Transfer between IF1/IF2 Register Set and Message RAM is in progress.<br>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.<br>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<br>0x00 Invalid message number<br>0x01-0x20 Valid message numbers<br>0x21-0xFF Invalid message numbers<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 22.15.2.23 CAN_IF1MSK Register (Offset = 104h) [Reset = FFFFFFFFh]

CAN_IF1MSK is shown in Figure 22-44 and described in Table 22-32.

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 22-44. 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 22-32. CAN_IF1MSK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | MXtd | R/W | 1h | Mask Extended Identifier<br>0 The extended identifier bit (Xtd) has no effect on the acceptance filtering.<br>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.<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |
| 30 | MDir | R/W | 1h | Mask Message Direction<br>0 The message direction bit (Dir) has no effect on the acceptance filtering.<br>1 The message direction bit (Dir) is used for acceptance filtering.<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |
| 29 | RESERVED | R | 1h | Reserved |
| 28-0 | Msk | R/W | 1FFFFFFFh | Identifier Mask-<br>0 The corresponding bit in the identifier of the message object is not used for acceptance filtering (don't care).<br>1 The corresponding bit in the identifier of the message object is used for acceptance filtering.<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 22.15.2.24 CAN_IF1ARB Register (Offset = 108h) [Reset = 00000000h]

CAN_IF1ARB is shown in Figure 22-45 and described in Table 22-33.

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 22-45. 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 22-33. CAN_IF1ARB Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | MsgVal | R/W | 0h | Message Valid<br>0 The mailbox is disabled. (The message object is ignored by the message handler).<br>1 The mailbox is enabled. (The message object is to be used by the message handler).<br>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.<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |
| 30 | Xtd | R/W | 0h | Extended Identifier<br>0 The 11-bit ('standard') Identifier is used for this message object.<br>1 The 29-bit ('extended') Identifier is used for this message object.<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 22-33. CAN_IF1ARB Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 29 | Dir | R/W | 0h | Message Direction<br>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.<br>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).<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |
| 28-0 | ID | R/W | 0h | Message Identifier<br>ID[28:0] 29-bit Identifier ('Extended Frame')<br>ID[28:18] 11-bit Identifier ('Standard Frame')<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 22.15.2.25 CAN_IF1MCTL Register (Offset = 10Ch) [Reset = 00000000h]

CAN_IF1MCTL is shown in Figure 22-46 and described in Table 22-34.

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 22-46. 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 22-34. CAN_IF1MCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15 | NewDat | R/W | 0h | New Data<br>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.<br>1 The message handler or the CPU has written new data into the data portion of this message object.<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |
| 14 | MsgLst | R/W | 0h | Message Lost (only valid for message objects with direction = receive)<br>0 No message lost since the last time when this bit was reset by the CPU.<br>1 The message handler stored a new message into this object when NewDat was still set, so the previous message has been overwritten.<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |
| 13 | IntPnd | R/W | 0h | Interrupt Pending<br>0 This message object is not the source of an interrupt.<br>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.<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |
| 12 | UMask | R/W | 0h | Use Acceptance Mask<br>0 Mask ignored<br>1 Use Mask (Msk[28:0], MXtd, and MDir) for acceptance filtering<br>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.<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 22-34. CAN_IF1MCTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 11 | TxIE | R/W | 0h | Transmit Interrupt Enable<br>0 IntPnd will not be triggered after the successful transmission of a frame.<br>1 IntPnd will be triggered after the successful transmission of a frame.<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |
| 10 | RxIE | R/W | 0h | Receive Interrupt Enable<br>0 IntPnd will not be triggered after the successful reception of a frame.<br>1 IntPnd will be triggered after the successful reception of a frame.<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |
| 9 | RmtEn | R/W | 0h | Remote Enable<br>0 At the reception of a remote frame, TxRqst is not changed.<br>1 At the reception of a remote frame, TxRqst is set.<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |
| 8 | TxRqst | R/W | 0h | Transmit Request<br>0 This message object is not waiting for a transmission.<br>1 The transmission of this message object is requested and is not yet done.<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |
| 7 | EoB | R/W | 0h | End of Block<br>0 The message object is part of a FIFO Buffer block and is not the last message object of the FIFO Buffer block.<br>1 The message object is a single message object or the last message object in a FIFO Buffer Block.<br>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.<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |
| 6-4 | RESERVED | R | 0h | Reserved |
| 3-0 | DLC | R/W | 0h | Data length code<br>0-8 Data frame has 0-8 data bytes.<br>9-15 Data frame has 8 data bytes.<br>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.<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 22.15.2.26 CAN_IF1DATA Register (Offset = 110h) [Reset = 00000000h]

CAN_IF1DATA is shown in Figure 22-47 and described in Table 22-35.

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 22-47. 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 22-35. CAN_IF1DATA Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-24 | Data_3 | R/W | 0h | Data Byte 3<br>Reset type: SYSRSn |
| 23-16 | Data_2 | R/W | 0h | Data Byte 2<br>Reset type: SYSRSn |
| 15-8 | Data_1 | R/W | 0h | Data Byte 1<br>Reset type: SYSRSn |
| 7-0 | Data_0 | R/W | 0h | Data Byte 0<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 22.15.2.27 CAN_IF1DATB Register (Offset = 114h) [Reset = 00000000h]

CAN_IF1DATB is shown in Figure 22-48 and described in Table 22-36.

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 22-48. 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 22-36. CAN_IF1DATB Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-24 | Data_7 | R/W | 0h | Data Byte 7<br>Reset type: SYSRSn |
| 23-16 | Data_6 | R/W | 0h | Data Byte 6<br>Reset type: SYSRSn |
| 15-8 | Data_5 | R/W | 0h | Data Byte 5<br>Reset type: SYSRSn |
| 7-0 | Data_4 | R/W | 0h | Data Byte 4<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 22.15.2.28 CAN_IF2CMD Register (Offset = 120h) [Reset = 00000001h]

CAN_IF2CMD is shown in Figure 22-49 and described in Table 22-37.

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 22-49. 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 22-37. CAN_IF2CMD Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-24 | RESERVED | R | 0h | Reserved |
| 23 | DIR | R/W | 0h | Write/Read<br>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.<br>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.<br>The other bits of IF1/IF2 Command Mask Register have different functions depending on the transfer direction.<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 22-37. CAN_IF2CMD Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 22 | Mask | R/W | 0h | Access Mask Bits<br>0 Mask bits will not be changed<br>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.<br>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]).<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |
| 21 | Arb | R/W | 0h | Access Arbitration Bits<br>0 Arbitration bits will not be changed<br>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.<br>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]).<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |
| 20 | Control | R/W | 0h | Access control bits.<br>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.<br>0 Control bits will not be changed.<br>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.<br>1 (Direction = Write): The message control bits will be transferred from the IF1 register set to the<br>message object addressed by message number (Bits [7:0]).<br>Note: This bit is write protected by the Busy bit.<br>Reset type: SYSRSn |
| 19 | ClrIntPnd | R/W | 0h | Clear Interrupt Pending Bit<br>0 IntPnd bit will not be changed<br>1 (Direction = Read): Clears IntPnd bit in the message object.<br>1 (Direction = Write): This bit is ignored.<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |
| 18 | TxRqst | R/W | 0h | Access Transmission Request (TxRqst) / New Data (NewDat) Bit<br>0 (Direction = Read): NewDat bit will not be changed.<br>0 (Direction = Write): TxRqst/NewDat bit will be handled according to the Control bit.<br>1 (Direction = Read): Clears NewDat bit in the message object.<br>1 (Direction = Write): Sets TxRqst/NewDat in message object.<br>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.<br>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.<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 22-37. CAN_IF2CMD Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 17 | DATA_A | R/W | 0h | Access Data Bytes 0-3<br>0 Data Bytes 0-3 will not be changed.<br>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.<br>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]).<br>Note: The duration of the message transfer is independent of the number of bytes to be transferred.<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |
| 16 | DATA_B | R/W | 0h | Access Data Bytes 4-7<br>0 Data Bytes 4-7 will not be changed.<br>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.<br>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]).<br>Note: The duration of the message transfer is independent of the number of bytes to be transferred.<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |
| 15 | Busy | R | 0h | Busy Flag<br>0 No transfer between IF1/IF2 Register Set and Message RAM is in progress.<br>1 Transfer between IF1/IF2 Register Set and Message RAM is in progress.<br>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.<br>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<br>0x00 Invalid message number<br>0x01-0x20 Valid message numbers<br>0x21-0xFF Invalid message numbers<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 22.15.2.29 CAN_IF2MSK Register (Offset = 124h) [Reset = FFFFFFFFh]

CAN_IF2MSK is shown in Figure 22-50 and described in Table 22-38.

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 22-50. 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 22-38. CAN_IF2MSK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | MXtd | R/W | 1h | Mask Extended Identifier<br>0 The extended identifier bit (Xtd) has no effect on the acceptance filtering.<br>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.<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |
| 30 | MDir | R/W | 1h | Mask Message Direction<br>0 The message direction bit (Dir) has no effect on the acceptance filtering.<br>1 The message direction bit (Dir) is used for acceptance filtering.<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |
| 29 | RESERVED | R | 1h | Reserved |
| 28-0 | Msk | R/W | 1FFFFFFFh | Identifier Mask<br>0 The corresponding bit in the identifier of the message object is not used for acceptance filtering (don't care).<br>1 The corresponding bit in the identifier of the message object is used for acceptance filtering.<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 22.15.2.30 CAN_IF2ARB Register (Offset = 128h) [Reset = 00000000h]

CAN_IF2ARB is shown in Figure 22-51 and described in Table 22-39.

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 22-51. 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 22-39. CAN_IF2ARB Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | MsgVal | R/W | 0h | Message Valid<br>0 The mailbox is disabled. (The message object is ignored by the message handler).<br>1 The mailbox is enabled. (The message object is to be used by the message handler).<br>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.<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |
| 30 | Xtd | R/W | 0h | Extended Identifier<br>0 The 11-bit ('standard') Identifier is used for this message object.<br>1 The 29-bit ('extended') Identifier is used for this message object.<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 22-39. CAN_IF2ARB Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 29 | Dir | R/W | 0h | Message Direction<br>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.<br>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).<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |
| 28-0 | ID | R/W | 0h | Message Identifier<br>ID[28:0] 29-bit Identifier ('Extended Frame')<br>ID[28:18] 11-bit Identifier ('Standard Frame')<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 22.15.2.31 CAN_IF2MCTL Register (Offset = 12Ch) [Reset = 00000000h]

CAN_IF2MCTL is shown in Figure 22-52 and described in Table 22-40.

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 22-52. 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 22-40. CAN_IF2MCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15 | NewDat | R/W | 0h | New Data<br>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.<br>1 The message handler or the CPU has written new data into the data portion of this message object.<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |
| 14 | MsgLst | R/W | 0h | Message Lost (only valid for message objects with direction = receive)<br>0 No message lost since the last time when this bit was reset by the CPU.<br>1 The message handler stored a new message into this object when NewDat was still set, so the previous message has been overwritten.<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |
| 13 | IntPnd | R/W | 0h | Interrupt Pending<br>0 This message object is not the source of an interrupt.<br>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.<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |
| 12 | UMask | R/W | 0h | Use Acceptance Mask<br>0 Mask ignored<br>1 Use Mask (Msk[28:0], MXtd, and MDir) for acceptance filtering<br>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.<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 22-40. CAN_IF2MCTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 11 | TxIE | R/W | 0h | Transmit Interrupt Enable<br>0 IntPnd will not be triggered after the successful transmission of a frame.<br>1 IntPnd will be triggered after the successful transmission of a frame.<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |
| 10 | RxIE | R/W | 0h | Receive Interrupt Enable<br>0 IntPnd will not be triggered after the successful reception of a frame.<br>1 IntPnd will be triggered after the successful reception of a frame.<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |
| 9 | RmtEn | R/W | 0h | Remote Enable<br>0 At the reception of a remote frame, TxRqst is not changed.<br>1 At the reception of a remote frame, TxRqst is set.<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |
| 8 | TxRqst | R/W | 0h | Transmit Request<br>0 This message object is not waiting for a transmission.<br>1 The transmission of this message object is requested and is not yet done.<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |
| 7 | EoB | R/W | 0h | End of Block<br>0 The message object is part of a FIFO Buffer block and is not the last message object of the FIFO Buffer block.<br>1 The message object is a single message object or the last message object in a FIFO Buffer Block.<br>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.<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |
| 6-4 | RESERVED | R | 0h | Reserved |
| 3-0 | DLC | R/W | 0h | Data length code<br>0-8 Data frame has 0-8 data bytes.<br>9-15 Data frame has 8 data bytes.<br>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.<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 22.15.2.32 CAN_IF2DATA Register (Offset = 130h) [Reset = 00000000h]

CAN_IF2DATA is shown in Figure 22-53 and described in Table 22-41.

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 22-53. 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 22-41. CAN_IF2DATA Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-24 | Data_3 | R/W | 0h | Data Byte 3<br>Reset type: SYSRSn |
| 23-16 | Data_2 | R/W | 0h | Data Byte 2<br>Reset type: SYSRSn |
| 15-8 | Data_1 | R/W | 0h | Data Byte 1<br>Reset type: SYSRSn |
| 7-0 | Data_0 | R/W | 0h | Data Byte 0<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 22.15.2.33 CAN_IF2DATB Register (Offset = 134h) [Reset = 00000000h]

CAN_IF2DATB is shown in Figure 22-54 and described in Table 22-42.

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 22-54. 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 22-42. CAN_IF2DATB Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-24 | Data_7 | R/W | 0h | Data Byte 7<br>Reset type: SYSRSn |
| 23-16 | Data_6 | R/W | 0h | Data Byte 6<br>Reset type: SYSRSn |
| 15-8 | Data_5 | R/W | 0h | Data Byte 5<br>Reset type: SYSRSn |
| 7-0 | Data_4 | R/W | 0h | Data Byte 4<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 22.15.2.34 CAN_IF3OBS Register (Offset = 140h) [Reset = 00000000h]

CAN_IF3OBS is shown in Figure 22-55 and described in Table 22-43.

Return to the Summary Table.

The IF3 register set can automatically be updated with received message objects without the need to initiate the transfer from Message RAM by CPU.
The observation flags (Bits [4:0]) in the IF3 Observation register are used to determine, which data sections of the IF3 Interface Register set have to be read in order to complete a DMA read cycle. After all marked data sections are read, the DCAN is enabled to update the IF3 Interface Register set with new data.
Any access order of single bytes or half-words is supported. When using byte or half-word accesses, a data section is marked as completed, if all bytes are read.
Note: If IF3 Update Enable is used and no Observation flag is set, the corresponding message objects will be copied to IF3 without activating the DMA request line and without waiting for DMA read accesses.
A write access to this register aborts a pending DMA cycle by resetting the DMA line and enables updating of IF3 Interface Register set with new data. To avoid data inconsistency, the DMA controller should be disabled before reconfiguring IF3 observation register. The status of the current read-cycle can be observed via status flags (Bits [12:8]).
With this, the observation status bits and the IF3Upd bit could be used by the application to realize the notification about new IF3 content in polling or interrupt mode

**Figure 22-55. CAN_IF3OBS Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| IF3Upd | RESERVED | | IF3SDB | IF3SDA | IF3SC | IF3SA | IF3SM |
| R-0h | R-0h | | R-0h | R-0h | R-0h | R-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | Data_B | Data_A | Ctrl | Arb | Mask |
| R-0h | | | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 22-43. CAN_IF3OBS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15 | IF3Upd | R | 0h | IF3 Update Data<br>0 No new data has been loaded since last IF3 read.<br>1 New data has been loaded since last IF3 read.<br>Reset type: SYSRSn |
| 14-13 | RESERVED | R | 0h | Reserved |
| 12 | IF3SDB | R | 0h | IF3 Status of Data B read access<br>0 All Data B bytes are already read out, or are not marked to be read.<br>1 Data B section has still data to be read out.<br>Reset type: SYSRSn |
| 11 | IF3SDA | R | 0h | IF3 Status of Data A read access<br>0 All Data A bytes are already read out, or are not marked to be read.<br>1 Data A section has still data to be read out.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 22-43. CAN_IF3OBS Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 10 | IF3SC | R | 0h | IF3 Status of Control bits read access<br>0 All Control section bytes are already read out, or are not marked to be read.<br>1 Control section has still data to be read out.<br>Reset type: SYSRSn |
| 9 | IF3SA | R | 0h | IF3 Status of Arbitration data read access<br>0 All Arbitration data bytes are already read out, or are not marked to be read.<br>1 Arbitration section has still data to be read out.<br>Reset type: SYSRSn |
| 8 | IF3SM | R | 0h | IF3 Status of Mask data read access<br>0 All Mask data bytes are already read out, or are not marked to be read.<br>1 Mask section has still data to be read out.<br>Reset type: SYSRSn |
| 7-5 | RESERVED | R | 0h | Reserved |
| 4 | Data_B | R/W | 0h | Data B read observation<br>0 Data B section not to be read.<br>1 Data B section has to be read to enable next IF3 update.<br>Reset type: SYSRSn |
| 3 | Data_A | R/W | 0h | Data A read observation<br>0 Data A section not to be read.<br>1 Data A section has to be read to enable next IF3 update.<br>Reset type: SYSRSn |
| 2 | Ctrl | R/W | 0h | Ctrl read observation<br>0 Ctrl section not to be read.<br>1 Ctrl section has to be read to enable next IF3 update.<br>Reset type: SYSRSn |
| 1 | Arb | R/W | 0h | Arbitration data read observation<br>0 Arbitration data not to be read.<br>1 Arbitration data has to be read to enable next IF3 update.<br>Reset type: SYSRSn |
| 0 | Mask | R/W | 0h | Mask data read observation<br>0 Mask data not to be read.<br>1 Mask data has to be read to enable next IF3 update.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 22.15.2.35 CAN_IF3MSK Register (Offset = 144h) [Reset = FFFFFFFFh]

CAN_IF3MSK is shown in Figure 22-56 and described in Table 22-44.

Return to the Summary Table.

This register provides a window to the acceptance mask for the chosen mailbox.

**Figure 22-56. 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 22-44. CAN_IF3MSK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | MXtd | R | 1h | Mask Extended Identifier<br>0 The extended identifier bit (Xtd) has no effect on the acceptance filtering.<br>1 The extended identifier bit (Xtd) is used for acceptance filtering.<br>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.<br>Reset type: SYSRSn |
| 30 | MDir | R | 1h | Mask Message Direction<br>0 The message direction bit (Dir) has no effect on the acceptance filtering.<br>1 The message direction bit (Dir) is used for acceptance filtering.<br>Reset type: SYSRSn |
| 29 | RESERVED | R | 1h | Reserved |
| 28-0 | Msk | R | 1FFFFFFFh | Identifier Mask Identifier Mask<br>0 The corresponding bit in the identifier of the message object is not used for acceptance filtering (don't care).<br>1 The corresponding bit in the identifier of the message object is used for acceptance filtering. Identifier Mask<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 22.15.2.36 CAN_IF3ARB Register (Offset = 148h) [Reset = 00000000h]

CAN_IF3ARB is shown in Figure 22-57 and described in Table 22-45.

Return to the Summary Table.

The bits of the IF3 Arbitration Register mirrors the arbitration bits of a message object.

**Figure 22-57. 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 22-45. CAN_IF3ARB Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | MsgVal | R | 0h | Message Valid<br>0 The message object is ignored by the message handler.<br>1 The message object is to be used by the message handler.<br>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.<br>Reset type: SYSRSn |
| 30 | Xtd | R | 0h | Extended Identifier<br>0 The 11-bit ('standard') Identifier is used for this message object.<br>1 The 29-bit ('extended') Identifier is used for this message object.<br>Reset type: SYSRSn |
| 29 | Dir | R | 0h | Message Direction<br>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.<br>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).<br>Reset type: SYSRSn |
| 28-0 | ID | R | 0h | Message Identifier<br>ID[28:0] 29-bit Identifier ('Extended Frame')<br>ID[28:18] 11-bit Identifier ('Standard Frame')<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 22.15.2.37 CAN_IF3MCTL Register (Offset = 14Ch) [Reset = 00000000h]

CAN_IF3MCTL is shown in Figure 22-58 and described in Table 22-46.

Return to the Summary Table.

The bits of the IF3 Message Control Register mirrors the message control bits of a message object.

**Figure 22-58. 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 22-46. CAN_IF3MCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15 | NewDat | R | 0h | New Data<br>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.<br>1 The message handler or the CPU has written new data into the data portion of this message object.<br>Reset type: SYSRSn |
| 14 | MsgLst | R | 0h | Message Lost (only valid for message objects with direction = receive)<br>0 No message lost since the last time when this bit was reset by the CPU.<br>1 The message handler stored a new message into this object when NewDat was still set, so the previous message has been overwritten.<br>Reset type: SYSRSn |
| 13 | IntPnd | R | 0h | Interrupt Pending<br>0 This message object is not the source of an interrupt.<br>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.<br>Reset type: SYSRSn |
| 12 | UMask | R | 0h | Use Acceptance Mask<br>0 Mask ignored<br>1 Use Mask (Msk[28:0], MXtd, and MDir) for acceptance filtering<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 22-46. CAN_IF3MCTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 11 | TxIE | R | 0h | Transmit Interrupt Enable<br>0 IntPnd will not be triggered after the successful transmission of a frame.<br>1 IntPnd will be triggered after the successful transmission of a frame.<br>Reset type: SYSRSn |
| 10 | RxIE | R | 0h | Receive Interrupt Enable<br>0 IntPnd will not be triggered after the successful reception of a frame.<br>1 IntPnd will be triggered after the successful reception of a frame.<br>Reset type: SYSRSn |
| 9 | RmtEn | R | 0h | Remote Enable<br>0 At the reception of a remote frame, TxRqst is not changed.<br>1 At the reception of a remote frame, TxRqst is set.<br>Reset type: SYSRSn |
| 8 | TxRqst | R | 0h | Transmit Request<br>0 This message object is not waiting for a transmission.<br>1 The transmission of this message object is requested and is not yet done.<br>Reset type: SYSRSn |
| 7 | EoB | R | 0h | End of Block<br>0 The message object is part of a FIFO Buffer block and is not the last message object of the FIFO Buffer block.<br>1 The message object is a single message object or the last message object in a FIFO Buffer Block.<br>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.<br>Reset type: SYSRSn |
| 6-4 | RESERVED | R | 0h | Reserved |
| 3-0 | DLC | R | 0h | Data length code<br>0-8 Data frame has 0-8 data bytes.<br>9-15 Data frame has 8 data bytes.<br>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.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 22.15.2.38 CAN_IF3DATA Register (Offset = 150h) [Reset = 00000000h]

CAN_IF3DATA is shown in Figure 22-59 and described in Table 22-47.

Return to the Summary Table.

This register provides a window to the data bytes of the CAN message.

**Figure 22-59. 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 22-47. CAN_IF3DATA Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-24 | Data_3 | R | 0h | Data Byte 3<br>Reset type: SYSRSn |
| 23-16 | Data_2 | R | 0h | Data Byte 2<br>Reset type: SYSRSn |
| 15-8 | Data_1 | R | 0h | Data Byte 1<br>Reset type: SYSRSn |
| 7-0 | Data_0 | R | 0h | Data Byte 0<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 22.15.2.39 CAN_IF3DATB Register (Offset = 154h) [Reset = 00000000h]

CAN_IF3DATB is shown in Figure 22-60 and described in Table 22-48.

Return to the Summary Table.

This register provides a window to the data bytes of the CAN message.

**Figure 22-60. 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 22-48. CAN_IF3DATB Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-24 | Data_7 | R | 0h | Data Byte 7<br>Reset type: SYSRSn |
| 23-16 | Data_6 | R | 0h | Data Byte 6<br>Reset type: SYSRSn |
| 15-8 | Data_5 | R | 0h | Data Byte 5<br>Reset type: SYSRSn |
| 7-0 | Data_4 | R | 0h | Data Byte 4<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 22.15.2.40 CAN_IF3UPD Register (Offset = 160h) [Reset = 00000000h]

CAN_IF3UPD is shown in Figure 22-61 and described in Table 22-49.

Return to the Summary Table.

The automatic update functionality of the IF3 register set can be configured for each message object. A message object is enabled for automatic IF3 update, if the dedicated IF3UpdEn flag is set. This means that an active NewDat flag of this message object (e.g due to reception of a CAN frame) will trigger an automatic copy of the whole message object to IF3 register set. Note: IF3 Update enable should not be set for transmit objects.

**Figure 22-61. 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 22-49. CAN_IF3UPD Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | IF3UpdEn | R/W | 0h | IF3 Update Enabled (for all message objects)<br>0 Automatic IF3 update is disabled for this message object.<br>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.<br>Reset type: SYSRSn |

### *22.15.3 CAN Registers to Driverlib Functions*

**Table 22-50. CAN Registers to Driverlib Functions**

| File | Driverlib Function |
|---|---|
| **CAN_CTL** | |
| 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 |
| **CAN_ES** | |
| can.c | CAN_clearInterruptStatus |
| can.h | CAN_getStatus |
| **CAN_ERRC** | |
| can.h | CAN_getErrorCount |
| **CAN_BTR** | |
| can.c | CAN_setBitTiming |



Copyright © 2024 Texas Instruments Incorporated

**Table 22-50. CAN Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| can.h | CAN_getBitTiming |
| **CAN_INT** | |
| can.h | CAN_getInterruptCause |
| **CAN_TEST** | |
| can.h | CAN_enableTestMode |
| can.h | CAN_disableTestMode |
| can.h | CAN_enableMemoryAccessMode |
| can.h | CAN_disableMemoryAccessMode |
| **CAN_PERR** | |
| - | |
| **CAN_RAM_INIT** | |
| can.h | CAN_initRAM |
| **CAN_GLB_INT_EN** | |
| can.h | CAN_enableGlobalInterrupt |
| can.h | CAN_disableGlobalInterrupt |
| **CAN_GLB_INT_FLG** | |
| can.h | CAN_getGlobalInterruptStatus |
| **CAN_GLB_INT_CLR** | |
| can.h | CAN_clearGlobalInterruptStatus |
| **CAN_ABOTR** | |
| can.h | CAN_setAutoBusOnTime |
| **CAN_TXRQ_X** | |
| - | |
| **CAN_TXRQ_21** | |
| can.h | CAN_getTxRequests |
| **CAN_NDAT_X** | |
| - | |
| **CAN_NDAT_21** | |
| can.h | CAN_getNewDataFlags |
| **CAN_IPEN_X** | |
| - | |
| **CAN_IPEN_21** | |
| can.h | CAN_getInterruptMessageSource |
| **CAN_MVAL_X** | |
| - | |
| **CAN_MVAL_21** | |
| can.h | CAN_getValidMessageObjects |
| **CAN_IP_MUX21** | |
| can.h | CAN_getInterruptMux |
| can.h | CAN_setInterruptMux |
| **CAN_IF1CMD** | |
| can.c | CAN_clearInterruptStatus |
| can.c | CAN_setupMessageObject |
| can.c | CAN_sendMessage |
| can.c | CAN_sendMessage_16bit |



Copyright © 2024 Texas Instruments Incorporated

**Table 22-50. 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 |
| **CAN_IF1MSK** | |
| can.c | CAN_setupMessageObject |
| **CAN_IF1ARB** | |
| can.c | CAN_setupMessageObject |
| can.c | CAN_clearMessage |
| can.c | CAN_disableMessageObject |
| can.c | CAN_disableAllMessageObjects |
| **CAN_IF1MCTL** | |
| 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 |
| **CAN_IF1DATA** | |
| can.c | CAN_sendMessage |
| can.c | CAN_sendMessage_16bit |
| can.c | CAN_sendMessage_32bit |
| can.c | CAN_sendMessage_updateDLC |
| **CAN_IF1DATB** | |
| - | See IF1DATA |
| **CAN_IF2CMD** | |
| can.c | CAN_readMessage |
| can.c | CAN_transferMessage |
| **CAN_IF2MSK** | |
| - | |
| **CAN_IF2ARB** | |
| can.c | CAN_readMessageWithID |
| **CAN_IF2MCTL** | |
| can.c | CAN_readMessage |
| **CAN_IF2DATA** | |
| can.c | CAN_readMessage |
| **CAN_IF2DATB** | |
| - | See IF2DATA |
| **CAN_IF3OBS** | |
| - | |
| **CAN_IF3MSK** | |
| - | |



Copyright © 2024 Texas Instruments Incorporated

**Table 22-50. CAN Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| **CAN_IF3ARB** | |
| - | |
| **CAN_IF3MCTL** | |
| - | |
| **CAN_IF3DATA** | |
| - | |
| **CAN_IF3DATB** | |
| - | See IF3DATA |
| **CAN_IF3UPD** | |
| - | |



Copyright © 2024 Texas Instruments Incorporated

Chapter 23

# Universal Serial Bus (USB) Controller

This chapter discusses the features and functions of the universal serial bus (USB) controller.


**23.1 Introduction** ..... 2636
**23.2 Functional Description** ..... 2639
**23.3 Initialization and Configuration** ..... 2650
**23.4 USB Global Interrupts** ..... 2651
**23.5 Software** ..... 2652
**23.6 USB Registers** ..... 2652





Copyright © 2024 Texas Instruments Incorporated

## 23.1 Introduction

The USB controller operates as a full-speed function controller during point-to-point communications with the USB host. The controller complies with the USB 2.0 standard, which includes SUSPEND and RESUME signaling. The USB controller has thirty-two endpoints, one-half of them being for IN transactions and one-half of them being for OUT transactions. One IN and one OUT endpoint are fixed-function endpoints used for control transfers; the others are defined by firmware. A dynamically sizeable FIFO supports queuing multiple packets. Software-controlled connect and disconnect allow flexibility during USB device startup.

### *23.1.1 Features*

The USB module has the following features:

- Complies with USB-IF certification standards
- USB 2.0 full-speed (12Mbps) operation in host and device modes as well as low-speed (1.5Mbps) operation in host mode
- Integrated PHY
- Three transfer types: Control, Interrupt, and Bulk
- 32 endpoints
  - One dedicated control IN endpoint and one dedicated control OUT endpoint
  - Fifteen configurable IN endpoints and fifteen configurable OUT endpoints
- 4KB dedicated endpoint memory

### *23.1.2 USB Related Collateral*

**Foundational Materials**

- C2000 Academy - USB
- USB Precision Labs (Video)

**Expert Materials**

- *High-Speed Interface Layout Guidelines Application Report*
- *USB Flash Programming of C2000 Microcontrollers Application Report*



Copyright © 2024 Texas Instruments Incorporated

### 23.1.3 Block Diagram

The USB block diagram is shown in Figure 23-1.

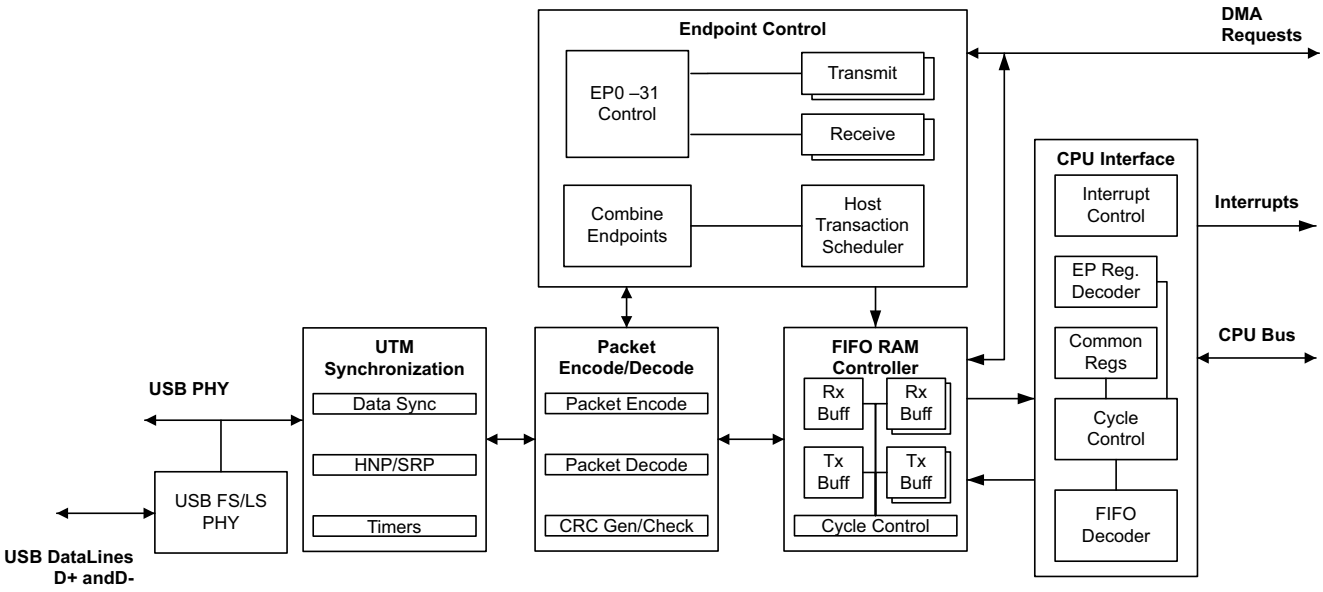



**Figure 23-1. USB Block Diagram**

#### 23.1.3.1 Signal Description

The USB controller requires a total of three signals (D+, D-, and $V_{Bus}$) to operate in device mode and two signals (D+, D-) to operate in embedded host mode. Because of the differential signaling needed for USB, the D+ and D- pins have special buffers to support USB. As such, the position on the chip is not user-selectable. The D+ and D- pins at reset are, by default, GPIOs and must be configured before being used as USB function pins. Bits 10 and 11 in the GPIO B Analog Mode Select register (GPBAMSEL) must be set to choose the USB function. The signals USB bus voltage ($V_{BUS}$), external power enable (EPEN), and power fault (PFLT) are not hardwired to any pin and some applications require these signals be implemented in software using a GPIO. Software that implements these signals is available in the USB software library.

#### 23.1.3.2 VBus Recommendations

Most applications do not need to monitor $V_{Bus}$. Because of this, a dedicated $V_{Bus}$ monitoring pin was not included on this microcontroller. If you are designing a bus-powered device application or an embedded host application, you do not need to monitor $V_{Bus}$. If you are designing a self-powered device, you need to actively monitor the state of the $V_{Bus}$ pin to make sure compliance with the USB specification. In Section 7.1.5 and Section 7.2.1 of the USB Specification Revision 2.0™:

- "The voltage source on the [speed identification] pull-up resistor must be derived from or controlled by the power supplied on the USB cable such that when $V_{BUS}$ is removed, the pull-up resistor does not supply current on the data line to which it is attached.
- When $V_{BUS}$ is removed, the device must remove power from the D+/D- pull-up resistor within 10 seconds.
- Later in the timing tables (Section 7.3.2) of the USB Specification 2.0, it is also stated that the D+/D- pull-up resistor must be applied within 100 ms of $V_{Bus}$ reaching a valid level."

Meeting the above specification is easy because of the slow timing requirements. The hardware part of the $V_{Bus}$ monitoring is discussed in this chapter. The corresponding software is discussed briefly, but for examples and an explanation, consult the USB software guide.

The pins of this microcontroller are not 5V tolerant, and because of this, the $V_{Bus}$ signal cannot be directly connected to a GPIO pin. Directly connecting 5V to a pin of the microcontroller destroys the I/O buffer of the pin



Copyright © 2024 Texas Instruments Incorporated

and possibly more of the chip. The most cost-effective way of making any pin capable of reading a 5V input is to use a series resistance in conjunction with the ESD diode clamps already present inside the device on every pin. Also, use a 100kΩ series resistor between the $V_{Bus}$ signal and the pin chosen to monitor the pin. A diagram of this setup is shown in Figure 23-2.

In Figure 23-2, if $V_{Bus}$ is above 3.3V or below 0V, one of the ESD clamp diodes is forward-biased, allowing current to flow through the 100kΩ resistor. The purpose of the diode clamps is to protect the pins of the microcontroller from very short over voltage spikes of a high magnitude. The diode clamps do this by clamping the voltage excursion to one of the supply rails. We are effectively requiring the ESD clamps to do the same thing the clamps were designed to do, but instead of a short high magnitude pulse, we are giving the clamps a long low magnitude static value using the 100kΩ resistor.

Any pin that has digital input and output functionality can potentially be used to monitor $V_{Bus}$, but the use of an interrupt-capable GPIO is recommended. A pin that does not have external interrupt capability can also be used, but the input state of the pin must be polled periodically by the application software to make sure appropriate action is taken whenever $V_{Bus}$ is applied or removed. If an interrupt-capable GPIO is chosen, the GPIO can be configured to generate an interrupt on both the rising and falling edge. More information on external interrupts can be found in the *System Control and Interrupts* chapter. Example code that implements $V_{Bus}$ monitoring using external interrupts and takes the appropriate actions is documented in the USB Software Guide and can be found in the associated USB software package.

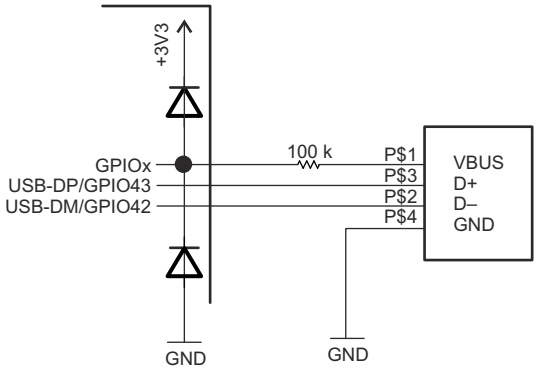



**Figure 23-2. USB Scheme**



Copyright © 2024 Texas Instruments Incorporated

## 23.2 Functional Description

The USB controller can be configured to act as either a dedicated host or device. However, when the USB controller is acting as a self-powered device, a GPIO input or analog comparator input must be connected to $V_{BUS}$ and configured to generate an interrupt when the $V_{BUS}$ level drops. This interrupt is used to disable the pullup resistor on the USB0DP signal.

**Note**

When a USB is used in the system, the minimum system frequency is 30MHz.

### *23.2.1 Operation as a Device*

This section describes how the USB controller performs when the USB controller is being used as a USB device. IN endpoints, OUT endpoints, entry into and exit from SUSPEND mode, and recognition of start of frame (SOF) are all described.

When in device mode, IN transactions are controlled by the endpoint transmit interface and uses the transmit endpoint registers for the given endpoint. OUT transactions are handled with the endpoints receive interface and use the receive endpoint registers for the given endpoint. When configuring the size of the FIFOs for endpoints, take into account the maximum packet size for an endpoint. Note the following:

- Bulk endpoints must be the size of the maximum packet (up to 64 bytes) or twice the maximum packet size if double buffering is used (described further in the following section).
- Interrupt endpoints must be the size of the maximum packet (up to 64 bytes) or twice the maximum packet size if double buffering is used.
- It is also possible to specify a separate control endpoint for a USB device. However, in most cases the USB device must use the dedicated control endpoint on the USB controller’s endpoint 0.

#### 23.2.1.1 Control and Configurable Endpoints

When operating as a device, the USB controller provides two dedicated control endpoints (IN and OUT). The remaining available configurable endpoints (one-half IN and one-half OUT) can be used for communications with a host controller. The endpoint number and direction associated with an endpoint is directly related to the register designation. For example, when the Host is transmitting to endpoint 1, all configuration and data is in the endpoint 1 transmit register interface. Endpoint 0 is a dedicated control endpoint used for all control transactions to endpoint 0 during enumeration or when any other control requests are made to endpoint 0. Endpoint 0 uses the first 64 bytes of the USB controller's FIFO RAM as a shared memory for both IN and OUT transactions.
The remaining six endpoints can be configured as control, bulk, or interrupt endpoints. The six endpoints can be treated as three configurable IN and three configurable OUT endpoints. The endpoint pairs are not required to have the same type for the IN and OUT endpoint configuration. For example, the OUT portion of an endpoint pair can be a bulk endpoint, while the IN portion of that endpoint pair can be an interrupt endpoint. The address and size of the FIFOs attached to each endpoint can be modified to fit the application's needs.



Copyright © 2024 Texas Instruments Incorporated

#### 23.2.1.1.1 IN Transactions as a Device

When operating as a USB device, data for IN transactions is handled through the FIFOs attached to the transmit endpoints. The sizes of the FIFOs for the configurable IN endpoints are determined by the USB Transmit FIFO Start Address (USBTXFIFOADD) register. The maximum size of a data packet that can be placed in a transmit endpoint's FIFO for transmission is programmable and is determined by the value written to the USB Maximum Transmit Data Endpoint n (USBTXMAXPn) register for that endpoint. The endpoint's FIFO can also be configured to use double-packet or single-packet buffering. When double-packet buffering is enabled, two data packets can be buffered in the FIFO, which also requires that the FIFO is at least two packets in size. When double-packet buffering is disabled, only one packet can be buffered, even if the packet size is less than half the FIFO size.

**Note**

The maximum packet size set for any endpoint must not exceed the FIFO size. The USBTXMAXPn register cannot be written to while data is in the FIFO as unexpected results can occur.

**Single-Packet Buffering**

If the size of the transmit endpoint's FIFO is less than twice the maximum packet size for this endpoint (as set in the USB Transmit Dynamic FIFO Sizing (USBTXFIFOSZ) register), only one packet can be buffered in the FIFO and single-packet buffering is required. When each packet is completely loaded into the transmit FIFO, the TXRDY bit in the USB Transmit Control and Status Endpoint n Low (USBTXCSRLn) register must be set. If the AUTOSET bit in the USB Transmit Control and Status Endpoint n High (USBTXCSRHn) register is set, the TXRDY bit is automatically set when a maximum-sized packet is loaded into the FIFO. For packet sizes less than the maximum, the TXRDY bit must be set manually. When the TXRDY bit is set, either manually or automatically, the packet is ready to be sent. When the packet has been successfully sent, both TXRDY and FIFONE are cleared, and the appropriate transmit endpoint interrupt signaled. At this point, the next packet can be loaded into the FIFO.

**Double-Packet Buffering**

If the size of the transmit endpoint's FIFO is at least twice the maximum packet size for this endpoint, two packets can be buffered in the FIFO and double-packet buffering is allowed. As each packet is loaded into the transmit FIFO, the TXRDY bit in the USBTXCSRLn register must be set. If the AUTOSET bit in the USBTXCSRHn register is set, the TXRDY bit is automatically set when a maximum-sized packet is loaded into the FIFO. For packet sizes less than the maximum, TXRDY must be set manually. When the TXRDY bit is set, either manually or automatically, the packet is ready to be sent. After the first packet is loaded, TXRDY is immediately cleared and an interrupt is generated. A second packet can now be loaded into the transmit FIFO and TXRDY set again (either manually or automatically if the packet is the maximum size). At this point, both packets are ready to be sent. After each packet has been successfully sent, TXRDY is automatically cleared and the appropriate transmit endpoint interrupt signaled to indicate that another packet can now be loaded into the transmit FIFO. The state of the FIFONE bit in the USBTXCSRLn register at this point indicates how many packets can be loaded. If the FIFONE bit is set, then another packet is in the FIFO and only one more packet can be loaded. If the FIFONE bit is clear, then no packets are in the FIFO and two more packets can be loaded.

**Note**

Double-packet buffering is disabled if an endpoint's corresponding EPn bit is set in the USB Transmit Double Packet Buffer Disable (USBTXDPKTBUFDIS) register. This bit is set by default, so the bit must be cleared to enable double-packet buffering.



Copyright © 2024 Texas Instruments Incorporated

#### 23.2.1.1.2 Out Transactions as a Device

When in device mode, OUT transactions are handled through the USB controller receive FIFOs. The sizes of the receive FIFOs for the configurable OUT endpoints are determined by the USB Receive FIFO Start Address (USBRXFIFOADD) register. The maximum amount of data received by an endpoint in any packet is determined by the value written to the USB Maximum Receive Data Endpoint n (USBRXMAXPn) register for that endpoint. When double-packet buffering is enabled, two data packets can be buffered in the FIFO. When double-packet buffering is disabled, only one packet can be buffered even if the packet is less than half the FIFO size.

**Note**

In all cases, the maximum packet size must not exceed the FIFO size.

**Single-Packet Buffering**

If the size of the receive endpoint FIFO is less than twice the maximum packet size for an endpoint, only one data packet can be buffered in the FIFO and single-packet buffering is required. When a packet is received and placed in the receive FIFO, the RXRDY and FULL bits in the USB Receive Control and Status Endpoint n Low (USBRXCSRL[*n*]) register are set and the appropriate receive endpoint is signaled, indicating that a packet can now be unloaded from the FIFO. After the packet has been unloaded, the RXRDY bit must be cleared to allow further packets to be received. This action also generates the acknowledge signaling to the Host controller. If the AUTOCL bit in the USB Receive Control and Status Endpoint n High (USBRXCSRH[*n*]) register is set and a maximum-sized packet is unloaded from the FIFO, the RXRDY and FULL bits are cleared automatically. For packet sizes less than the maximum, RXRDY must be cleared manually.

**Double-Packet Buffering**

If the size of the receive endpoint FIFO is at least twice the maximum packet size for the endpoint, two data packets can be buffered and double-packet buffering can be used. When the first packet is received and loaded into the receive FIFO, the RXRDY bit in the USBRXCSRL[*n*] register is set and the appropriate receive endpoint interrupt is signaled to indicate that a packet can now be unloaded from the FIFO.

**Note**

The FULL bit in USBRXCSRL[*n*] is not set when the first packet is received. The FULL bit is only set if a second packet is received and loaded into the receive FIFO.

After each packet has been unloaded, the RXRDY bit must be cleared to allow further packets to be received. If the AUTOCL bit in the USBRXCSRH[*n*] register is set and a maximum-sized packet is unloaded from the FIFO, the RXRDY bit is cleared automatically. For packet sizes less than the maximum, RXRDY must be cleared manually. If the FULL bit is set when RXRDY is cleared, the USB controller first clears the FULL bit, then sets RXRDY again to indicate that there is another packet waiting in the FIFO to be unloaded.

**Note**

Double-packet buffering is disabled if an endpoint's corresponding EPn bit is set in the USB Receive Double Packet Buffer Disable (USBRXDPKTBUFDIS) register. This bit is set by default, so the bit must be cleared to enable double-packet buffering.



Copyright © 2024 Texas Instruments Incorporated

#### 23.2.1.1.3 Scheduling

The device has no control over the scheduling of transactions as scheduling is determined by the Host controller. The USB controller can set up a transaction at any time. The USB controller waits for the request from the Host controller and generates an interrupt when the transaction is complete or if it was terminated due to some error. If the Host controller makes a request and the device controller is not ready, the USB controller sends a busy response (NAK) to all requests until it is ready.

#### 23.2.1.1.4 Additional Actions

The USB controller responds automatically to certain conditions on the USB bus or actions by the Host controller such as when the USB controller automatically stalls a control transfer or unexpected zero length OUT data packets.

**Stalled Control Transfer**

The USB controller automatically issues a STALL handshake to a control transfer under the following conditions:

1. The Host sends more data during an OUT data phase of a control transfer than was specified in the device request during the SETUP phase. This condition is detected by the USB controller when the Host sends an OUT token (instead of an IN token) after the last OUT packet has been unloaded and the DATAEND bit in the USB Control and Status Endpoint 0 Low (USBCSRL0) register has been set.
2. The Host requests more data during an IN data phase of a control transfer than was specified in the device request during the SETUP phase. This condition is detected by the USB controller when the Host sends an IN token (instead of an OUT token) after the CPU has cleared TXRDY and set DATAEND in response to the ACK issued by the Host to what must have been the last packet.
3. The Host sends more than USBRXMAXPn bytes of data with an OUT data token.
4. The Host sends more than a zero length data packet for the OUT STATUS phase.

**Zero Length OUT Data Packets**

A zero-length OUT data packet is used to indicate the end of a control transfer. In normal operation, such packets must only be received after the entire length of the device request has been transferred. However, if the Host sends a zero-length OUT data packet before the entire length of device request has been transferred, the Host is signaling the premature end of the transfer. In this case, the USB controller automatically flushes any IN token ready for the data phase from the FIFO and sets the DATAEND bit in the USBCSRL0 register.

**Setting the Device Address**

When a Host is attempting to enumerate the USB device, the Host requests that the device change the address from zero to some other value. The address is changed by writing the value that the Host requested to the USB Device Functional Address (USBFADDR) register. However, care must be taken when writing to USBFADDR to avoid changing the address before the transaction is complete. This register must only be set after the SET_ADDRESS command is complete. Like all control transactions, the transaction is only complete after the device has left the STATUS phase. In the case of a SET_ADDRESS command, the transaction is completed by responding to the IN request from the Host with a zero-byte packet. Once the device has responded to the IN request, the USBFADDR register must be programmed to the new value as soon as possible to avoid missing any new commands sent to the new address.

**Note**

If the USBFADDR register is set to the new value as soon as the device receives the OUT transaction with the SET_ADDRESS command in the packet, it changes the address during the control transfer. In this case, the device does not receive the IN request that allows the USB transaction to exit the STATUS phase of the control transfer because it is sent to the old address. As a result, the Host does not get a response to the IN request, and the Host fails to enumerate the device.



Copyright © 2024 Texas Instruments Incorporated

#### 23.2.1.1.5 Device Mode Suspend

When no activity has occurred on the USB bus for 3 ms, the USB controller automatically enters SUSPEND mode. If the SUSPEND interrupt has been enabled in the USB Interrupt Enable (USBIE) register, an interrupt is generated at this time. When in SUSPEND mode, the PHY also goes into SUSPEND mode. When RESUME signaling is detected, the USB controller exits SUSPEND mode and takes the PHY out of SUSPEND. If the RESUME interrupt is enabled, an interrupt is generated. The USB controller can also be forced to exit SUSPEND mode by setting the RESUME bit in the USB Power (USBPOWER) register. When this bit is set, the USB controller exits SUSPEND mode and drives RESUME signaling onto the bus. The RESUME bit must be cleared after 10 ms (a maximum of 15 ms) to end RESUME signaling. To meet USB power requirements, the controller can be put into Deep Sleep mode which keeps the controller in a static state.

#### 23.2.1.1.6 Start of Frame

When the USB controller is operating in device mode, it receives a Start-Of-Frame (SOF) packet from the Host once every millisecond. When the SOF packet is received, the 11-bit frame number contained in the packet is written into the USB Frame Value (USBFRAME) register, and an SOF interrupt is also signaled and can be handled by the application. Once the USB controller has started to receive SOF packets, it expects one every millisecond. If no SOF packet is received after 1.00358 ms, the packet is assumed to have been lost, and the USBFRAME register is not updated. The USB controller continues and resynchronizes these pulses to the received SOF packets when these packets are successfully received again.

#### 23.2.1.1.7 USB Reset

When the USB controller is in device mode and a RESET condition is detected on the USB bus, the USB controller automatically performs the following actions:

- Clears the USBFADDR register
- Clears the USB Endpoint Index (USBEPIDX) register
- Flushes all endpoint FIFOs
- Clears all control/status registers
- Enables all endpoint interrupts
- Generates a RESET interrupt

#### 23.2.1.1.8 Connect/Disconnect

The USB controller connection to the USB bus is handled by software. The USB PHY can be switched between normal mode and non-driving mode by setting or clearing the SOFTCONN bit of the USBPOWER register. When the SOFTCONN bit is set, the PHY is placed in the normal mode, and the USB0DP/USB0DM lines of the USB bus are enabled. At the same time, the USB controller is placed into a state that does not respond to any USB signaling except a USB RESET. When the SOFTCONN bit is cleared, the PHY is put into non-driving mode, USB0DP and USB0DM are tristated, and the USB controller appears to other devices on the USB bus as if the USB controller has been disconnected. The non-driving mode is the default so the USB controller appears disconnected until the SOFTCONN bit has been set. The application software can then choose when to set the PHY into the normal mode. Systems with a lengthy initialization procedure can use this to make sure that initialization is complete, and the system is ready to perform enumeration before connecting to the USB bus. Once the SOFTCONN bit has been set, the USB controller can be disconnected by clearing this bit.

**Note**

The USB controller does not generate an interrupt when the device is connected to the Host. However, an interrupt is generated when the Host terminates a session.



Copyright © 2024 Texas Instruments Incorporated

### 23.2.2 Operation as a Host

When the USB controller is operating in Host mode, the USB controller can either be used for point-to-point communications with another USB device or, when attached to a hub, for communication with multiple devices. Full-speed and low-speed USB devices are supported, both for point-to-point communication and for operation through a hub. The USB controller automatically carries out the necessary transaction translation needed to allow a low-speed or full-speed device to be used with a USB 2.0 hub. Control, bulk, and interrupt transactions are supported. This section describes the USB controller's actions when the USB controller is being used as a USB Host. Configuration of IN endpoints, OUT endpoints, entry into and exit from SUSPEND mode, and RESET are all described.

When in Host mode, IN transactions are controlled by an endpoint's receive interface. All IN transactions use the receive endpoint registers and all OUT endpoints use the transmit endpoint registers for a given endpoint. As in device mode, the FIFOs for endpoints must take into account the maximum packet size for an endpoint.

- Bulk endpoints must be the size of the maximum packet (up to 64 bytes) or twice the maximum packet size if double buffering is used (described further in the following section).
- Interrupt endpoints must be the size of the maximum packet (up to 64 bytes) or twice the maximum packet size if double buffering is used.
- It is also possible to specify a separate control endpoint to communicate with a device. However, in most cases the USB controller must use the dedicated control endpoint to communicate with a device's endpoint 0.

#### 23.2.2.1 Endpoint Registers

The endpoint registers are used to control the USB endpoint interfaces which communicate with devices that are connected. The endpoints consist of a dedicated control IN endpoint and a dedicated control OUT endpoint. The remaining available endpoints are configurable, with one-half of them being OUT endpoints, and one-half of them being IN endpoints. See Section 23.1.1 for the number of available endpoints on this device.

The dedicated control interface can only be used for control transactions to endpoint 0 of devices. These control transactions are used during enumeration or other control functions that communicate using endpoint 0 of devices. This control endpoint shares the first 64 bytes of the USB controller's FIFO RAM for IN and OUT transactions. The remaining IN and OUT interfaces can be configured to communicate with control, bulk, or interrupt endpoints.

These USB interfaces can be used to simultaneously schedule as many as 15 independent OUT and 15 independent IN transactions to any endpoints on any device. The IN and OUT controls are paired together in the same set of registers for the respective endpoints. However, the IN and OUT controls can be configured to communicate with different types of endpoints and different endpoints on devices. For example, the first pair of endpoint controls can be split so that the OUT portion is communicating with a device's bulk OUT endpoint 1, while the IN portion is communicating with a device's interrupt IN endpoint 2.

Before accessing any device, whether for point-to-point communications or for communications using a hub, the relevant USB Receive Functional Address Endpoint n (USBRXFUNCADDRn) or USB Transmit Functional Address Endpoint n (USBTXFUNCADDRn) registers must be set for each receive or transmit endpoint to record the address of the device being accessed.

The USB controller also supports connections to devices through a USB hub by providing a register that specifies the hub address and port of each USB transfer. The FIFO address and size are customizable and can be specified for each USB IN and OUT transfer. Customization includes allowing one FIFO per transaction, sharing a FIFO across transactions, and allowing for double-buffered FIFOs.



Copyright © 2024 Texas Instruments Incorporated

#### 23.2.2.2 IN Transactions as a Host

IN transactions are handled in a similar manner to the way in which OUT transactions are handled when the USB controller is in device mode except that the transaction first must be initiated by setting the REQPKT bit in the USBCSRL0 register, indicating to the transaction scheduler that there is an active transaction on this endpoint. The transaction scheduler then sends an IN token to the target device. When the packet is received and placed in the receive FIFO, the RXRDY bit in the USBCSRL0 register is set, and the appropriate receive endpoint interrupt is signaled to indicate that a packet can now be unloaded from the FIFO.

When the packet has been unloaded, RXRDY must be cleared. The AUTOCL bit in the USBRXCSRHn register can be used to have RXRDY automatically cleared when a maximum-sized packet has been unloaded from the FIFO. The AUTORQ bit in USBRXCSRHn causes the REQPKT bit to be automatically set when the RXRDY bit is cleared. When the RXRDY bit is cleared, the controller sends an acknowledge to the device. When there is a known number of packets to be transferred, the USB Request Packet Count in Block Transfer Endpoint n (USBRQPKTCOUNTn) register associated with the endpoint must be configured to the number of packets to be transferred. The USB controller decrements the value in the USBRQPKTCOUNTn register following each request. When the USBRQPKTCOUNTn value decrements to 0, the AUTORQ bit is cleared to prevent any further transactions being attempted. For cases where the size of the transfer is unknown, USBRQPKTCOUNTn must be cleared. AUTORQ then remains set until cleared by the reception of a short packet (that is, less than the MAXLOAD value in the USBRXMAXPn register) such as can occur at the end of a bulk transfer.

If the device responds to a bulk or interrupt IN token with a NAK, the USB Host controller keeps retrying the transaction until any NAK Limit that has been set has been reached. If the target device responds with a STALL, however, the USB Host controller does not retry the transaction but sets the STALLED bit in the USBCSRL0 register. If the target device does not respond to the IN token within the required time, or the packet contained a CRC or bit-stuff error, the USB Host controller retries the transaction. If after three attempts the target device has still not responded, the USB Host controller clears the REQPKT bit and sets the ERROR bit in the USBCSRL0 register.

#### 23.2.2.3 OUT Transactions as a Host

OUT transactions are handled in a similar manner to the way in which IN transactions are handled when the USB controller is in device mode. The TXRDY bit in the USBTXCSRLn register must be set as each packet is loaded into the transmit FIFO. Again, setting the AUTOSET bit in the USBTXCSRHn register automatically sets TXRDY when a maximum-sized packet has been loaded into the FIFO.

If the target device responds to the OUT token with a NAK, the USB Host controller keeps retrying the transaction until the NAK Limit that has been set has been reached. However, if the target device responds with a STALL, the USB controller does not retry the transaction but interrupts the main processor by setting the STALLED bit in the USBTXCSRLn register. If the target device does not respond to the OUT token within the required time, or the packet contained a CRC or bit-stuff error, the USB Host controller retries the transaction. If after three attempts the target device has still not responded, the USB controller flushes the FIFO and sets the ERROR bit in the USBTXCSRLn register.



Copyright © 2024 Texas Instruments Incorporated

#### 23.2.2.4 Transaction Scheduling

Scheduling of transactions is handled automatically by the USB Host controller. The Host controller allows configuration of the endpoint communication scheduling based on the type of endpoint transaction. Interrupt transactions can be scheduled to occur in the range of every frame to every 255 frames in 1 frame increments. Bulk endpoints do not allow scheduling parameters, but do allow for a NAK timeout in the event an endpoint on a device is not responding.

The USB controller maintains a frame counter. If the target device is a full-speed device, the USB controller automatically sends an SOF packet at the start of each frame and increments the frame counter. If the target device is a low-speed device, a K state is transmitted on the bus to act as a keep-alive to stop the low-speed device from going into SUSPEND mode.

After the SOF packet has been transmitted, the USB Host controller cycles through all the configured endpoints looking for active transactions. An active transaction is defined as a receive endpoint for which the REQPKT bit is set or a transmit endpoint for which the TXRDY bit and/or the FIFONE bit is set.

An interrupt transaction is started if the transaction is found on the first scheduler cycle of a frame and if the interval counter for that endpoint has counted down to zero. As a result, only one interrupt transaction occurs per endpoint every n frames, where n is the interval set using the USB Host Transmit Interval Endpoint n (USBTXINTERVAL[*n*]) or USB Host Receive Interval Endpoint n (USBRXINTERVAL[*n*]) register for that endpoint.

An active bulk transaction starts immediately, provided sufficient time is left in the frame to complete the transaction before the next SOF packet is due. If the transaction must be retried (for example, because a NAK was received or the target device did not respond), then the transaction is not retried until the transaction scheduler has first checked all the other endpoints for active transactions. This process makes sure that an endpoint that is sending a lot of NAKs does not block other transactions on the bus. The controller also allows the user to specify a limit to the length of time for NAKs to be received from a target device before the endpoint times out.



Copyright © 2024 Texas Instruments Incorporated

#### 23.2.2.5 USB Hubs

The following setup requirements apply to the USB Host controller only if it is used with a USB hub. When a full- or low-speed device is connected to the USB controller using a USB 2.0 hub, details of the hub address and the hub port also must be recorded in the corresponding USB Receive Hub Address Endpoint n (USBRXHUBADDRn) and USB Receive Hub Port Endpoint n (USBRXHUBPORTn) or the USB Transmit Hub Address Endpoint n (USBTXHUBADDRn) and USB Transmit Hub Port Endpoint n (USBTXHUBPORTn) registers. In addition, the speed at which the device operates (full or low) must be recorded in the USB Type Endpoint 0 (USBTYPE0) (endpoint 0), USB Host Configure Transmit Type Endpoint n (USBTXTYPEn), or USB Host Configure Receive Type Endpoint n (USBRXTYPEn) registers for each endpoint that is accessed by the device.

For hub communications, the settings in these registers record the current allocation of the endpoints to the attached USB devices. To maximize the number of devices supported, the USB Host controller allows this allocation to be changed dynamically by simply updating the address and speed information recorded in these registers. Any changes in the allocation of endpoints to device functions must be made following the completion of any on-going transactions on the endpoints affected.

#### 23.2.2.6 Babble

The USB Host controller does not start a transaction until the bus has been inactive for at least the minimum inter-packet delay. The controller also does not start a transaction unless it can be finished before the end of the frame. If the bus is still active at the end of a frame, then the USB Host controller assumes that the target device to which it is connected has malfunctioned, and the USB controller suspends all transactions and generates a babble interrupt.

#### 23.2.2.7 Host SUSPEND

If the SUSPEND bit in the USBPOWER register is set, the USB Host controller completes the current transaction then stops the transaction scheduler and frame counter. No further transactions are started and no SOF packets are generated.

To exit SUSPEND mode, set the RESUME bit and clear the SUSPEND bit. While the RESUME bit is set, the USB Host controller generates RESUME signaling on the bus. After 20ms, the RESUME bit must be cleared, at which point the frame counter and transaction scheduler start. The Host supports the detection of a remote wake-up.

#### 23.2.2.8 USB RESET

If the RESET bit in the USBPOWER register is set, the USB Host controller generates USB RESET signaling on the bus. The RESET bit must be set for at least 20 ms to make sure of correct resetting of the target device. After the CPU has cleared the bit, the USB Host controller starts the frame counter and transaction scheduler.

#### 23.2.2.9 Connect/Disconnect

A session is started by setting the SESSION bit in the USB device Control (USBDEVCTL) register, enabling the USB controller to wait for a device to be connected. When a device is detected, a connect interrupt is generated. The speed of the device that has been connected can be determined by reading the USBDEVCTL register where the FSDEV bit is set for a full-speed device, and the LSDEV bit is set for a low-speed device. The USB controller must generate a RESET to the device, and then the USB Host controller can begin device enumeration. If the device is disconnected while a session is in progress, a disconnect interrupt is generated.



Copyright © 2024 Texas Instruments Incorporated

### 23.2.3 DMA Operation

The USB module DMA trigger signals are not supported on this device. The DMA controller can be used to read and write the USB FIFOs using software triggering. See the *Direct Memory Access (DMA)* chapter for more details about programming the DMA controller. See the *USB DMA Event Trigger* advisory in the device errata for more information.

### 23.2.4 Address/Data Bus Bridge

This USB controller was originally designed to connect to an ARM AHB bus, but has been modified to function with the C28x device bus architecture. The modifications made are largely invisible to the user application, but there are some things to note.

- The USB memory space is 8 bits wide, while the C28x memory space is 16 bits wide.
- 32- and 16-bit accesses (r/w) are completely transparent to the user application code, no changes need be made.
- The C28x core only supports 8 bit accesses through a byte intrinsic type. This can be used to perform 8 bit reads or writes to the USB controller.
  - int &__byte(int *array, unsigned int byte_index);
  - *array = ptr to address to access, byte_index = always 0 (for USB)
    See Table 23-1 for example.
  - See the *TMS320C28x Optimizing C/C++ Compiler User's Guide* and the *TMS320C28x Assembly Language Tools User's Guide*
- Because of the bridge, the memory view of the USB controller memory space in CCS is not a 1:1 representation of what is in the controller
  - When the view mode is
    - 32 bit or 16 bit, even address are effectively duplicated, ignore odd addresses.
    - 8 bit, even addresses from within the controller are duplicated into odd address in the view window; old addresses from within the controller are not displayed.
      See Table 23-2 for example.



Copyright © 2024 Texas Instruments Incorporated

**Table 23-1. USB Memory Access from Software**

| USB Controller Memory | | | C28x 8 Bit | |
|---|---|---|---|---|
| **Address** | **Register Name** | **Data** | **Access** | **Data** |
| 0x00 | FADDR | 0x00 | __byte((int *)0x00,0) | 0x0000 |
| 0x01 | POWER | 0x11 | __byte((int *)0x01,0) | 0x0011 |
| 0x02 | TXIS (LSB) | 0x22 | __byte((int *)0x02,0) | 0x0022 |
| 0x03 | TXIS (MSB) | 0x33 | __byte((int *)0x03,0) | 0x0033 |
| 0x04 | RXIS (LSB) | 0x44 | __byte((int *)0x04,0) | 0x0044 |
| 0x05 | RXIS (MSB) | 0x55 | __byte((int *)0x05,0) | 0x0055 |
| 0x06 | TXIE (LSB) | 0x66 | __byte((int *)0x06,0) | 0x0066 |
| 0x07 | TXIE (MSB) | 0x77 | __byte((int *)0x07,0) | 0x0077 |
| 0x08 | RXIE (LSB) | 0x88 | __byte((int *)0x08,0) | 0x0088 |
| 0x09 | RXIE (MSB) | 0x99 | __byte((int *)0x09,0) | 0x0099 |
| 0x0A | USBIS | 0xAA | __byte((int *)0x0A,0) | 0x00AA |
| 0x0B | USBIE | 0xBB | __byte((int *)0x0B,0) | 0x00BB |
| 0x0C | FRAME (LSB) | 0xCC | __byte((int *)0x0C,0) | 0x00CC |
| 0x0D | FRAME (MSB) | 0xDD | __byte((int *)0x0D,0) | 0x00DD |
| 0x0E | EPIDX | 0xEE | __byte((int *)0x0E,0) | 0x00EE |
| 0x0F | TEST | 0xFF | __byte((int *)0x0F,0) | 0x00FF |
| **C28x 16 Bit** | | | **C28x 32 Bit** | |
| **Access** | **Data** | | **Access** | **Data** |
| (*((short *)(0x00))) | 0x1100 | | (*((long *)(0x00))) | 0x33221100 |
| (*((short *)(0x01))) | 0x1100 | | (*((long *)(0x01))) | 0x33221100 |
| (*((short *)(0x02))) | 0x3322 | | (*((long *)(0x02))) | 0x33221100 |
| (*((short *)(0x03))) | 0x3322 | | (*((long *)(0x03))) | 0x33221100 |
| (*((short *)(0x04))) | 0x5544 | | (*((long *)(0x04))) | 0x77665544 |
| (*((short *)(0x05))) | 0x5544 | | (*((long *)(0x05))) | 0x77665544 |
| (*((short *)(0x06))) | 0x7766 | | (*((long *)(0x06))) | 0x77665544 |
| (*((short *)(0x07))) | 0x7766 | | (*((long *)(0x07))) | 0x77665544 |
| (*((short *)(0x08))) | 0x9988 | | (*((long *)(0x08))) | 0xBBAA9988 |
| (*((short *)(0x09))) | 0x9988 | | (*((long *)(0x09))) | 0xBBAA9988 |
| (*((short *)(0x0A))) | 0xBBAA | | (*((long *)(0x0A))) | 0xBBAA9988 |
| (*((short *)(0x0B))) | 0xBBAA | | (*((long *)(0x0B))) | 0xBBAA9988 |
| (*((short *)(0x0C))) | 0xDDCC | | (*((long *)(0x0C))) | 0xFFEEDDCC |
| (*((short *)(0x0D))) | 0xDDCC | | (*((long *)(0x0D))) | 0xFFEEDDCC |
| (*((short *)(0x0E))) | 0xFFEE | | (*((long *)(0x0E))) | 0xFFEEDDCC |
| (*((short *)(0x0F))) | 0xFFEE | | (*((long *)(0x0F))) | 0xFFEEDDCC |



Copyright © 2024 Texas Instruments Incorporated

**Table 23-2. USB Memory Access from CCS 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 | | | |

## 23.3 Initialization and Configuration

To use the USB controller, the peripheral clock must be enabled using the System Control module PCLKCR11 register. In addition, the USB PHY signals must be connected to the respective pins using the GPIO module GPBAMSEL register. Set bits 10 and 11 for USB0DM (GPIO42) and USB0DP (GPIO43).

Set up the auxiliary PLL so a 60MHz output clock is provided to the USB module. This fixed frequency is required for all USB operations. See the *System Control and Interrupts* chapter for more details.

In host mode, the USB controller is responsible for supplying power to the bus. To avoid incorrectly supplying voltage to the bus, the external power control signal, USB0EPEN, must be kept inactive on start-up. This can be done by connecting the USB0EPEN and USB0PFLT pins to the USB controller as soon as possible.

### *23.3.1 Pin Configuration*

To give more flexibility, the signals External Power Enable (EPEN) and Power Fault (PFLT) were not implemented in hardware and the user must implement these signals in software. Examples of how to implement these signals in software can be found in the USB Software Guide located in C2000Ware in the \libraries\communications\usb\ directory.

When using the device controller portion of the USB controller in a system that also provides host functionality, the power to $V_{BUS}$ must be disabled to allow the external host controller to supply power. Usually, the EPEN signal is used to control the external regulator and must be negated to avoid having two devices driving the $V_{BUS}$ power pin on the USB connector.

When the USB controller is acting as a host, the USB controller is in control of two signals that are attached to an external voltage supply that provides power to $V_{BUS}$. The Host controller uses the EPEN signal to enable or disable power to the $V_{BUS}$ pin on the USB connector. An input pin, PFLT, provides feedback when there has been a power fault on $V_{BUS}$. The PFLT signal can be configured to either automatically negate the EPEN signal to disable power, or the PFLT signal can generate an interrupt to the interrupt controller to allow software to handle the power fault condition. The polarity and actions related to both EPEN and PFLT are fully configurable in the USB controller. The controller also provides interrupts on device insertion and removal to allow the Host controller code to respond to these external events.



Copyright © 2024 Texas Instruments Incorporated

### 23.3.2 Endpoint Configuration

To start communication in Host or device mode, the endpoint registers must first be configured. In Host mode, this configuration establishes a connection between an endpoint register and an endpoint on a device. In device mode, an endpoint must be configured before enumerating to the Host controller.

In both cases, the endpoint 0 configuration is limited because the endpoint is a fixed-function, fixed-FIFO-size endpoint. In device and Host modes, the endpoint requires little setup but does require a software-based state machine to progress through the setup, data, and status phases of a standard control transaction. In device mode, the configuration of the remaining endpoints is done once before enumerating and then only changed if an alternate configuration is selected by the Host controller. In Host mode, the endpoints must be configured to operate as control, bulk, or interrupt mode. Once the type of endpoint is configured, a FIFO area must be assigned to each endpoint. In the case of bulk, control and interrupt endpoints, each has a maximum of 64 bytes per transaction. The maximum packet size for the given endpoint must be set prior to sending or receiving data.

Configuring each endpoint's FIFO involves reserving a portion of the overall USB FIFO RAM to each endpoint. The total FIFO RAM available is 4 Kbytes with the first 64 bytes reserved for endpoint 0. The endpoint's FIFO must be at least as large as the maximum packet size. The FIFO can also be configured as a double-buffered FIFO so that interrupts occur at the end of each packet and allow filling the other half of the FIFO.

If operating as a device, the USB device controller's soft connect must be enabled when the device is ready to start communications, indicating to the host controller that the device is ready to start the enumeration process. If operating as a Host controller, the device soft connect must be disabled and power must be provided to $V_{BUS}$ using the USB0EPEN signal.

## 23.4 USB Global Interrupts

Global interrupt enable, flag, and clear registers have been added to make sure that no interrupt is missed. The USB interrupt can be enabled or blocked using the INTEN bit. The INTFLG bit indicates whether an interrupt has occurred or not. Finally, the INTFLGCLR bit clears the INTFLG when a value of 1 is written to the field.



Copyright © 2024 Texas Instruments Incorporated

## 23.5 Software

### 23.5.1 USB Examples

NOTE: These examples are located in the C2000Ware installation at the following location: C2000Ware_VERSION#/driverlib/DEVICE_GPN/examples/CORE_IF_MULTICORE/usb

Cloud access to these examples is available at the following link: dev.ti.com C2000Ware Examples.

## 23.6 USB Registers

This section describes the Universal Serial Bus (USB) Controller registers.

### 23.6.1 USB Base Address

**Table 23-3. USB Base Address Table (C28)**

| DriverLib Name | Base Address | CPU1 | CPU2 | DMA | CLA | Pipeline Protected |
|---|---|---|---|---|---|---|
| USB_BASE | 0x0004_0000 | YES | - | - | - | YES |

### 23.6.2 USB Register Map

Table 23-4 lists the registers. All addresses given are relative to the USB base address of 0x0000_5E00.

**Note**

The USB controller clock must be enabled before the registers can be programmed (see the *System Control and Interrupt* chapter).

**Table 23-4. Universal Serial Bus (USB) Controller Register Map**

| Offset | Acronym | Register Name | Section |
|---|---|---|---|
| 0x000 | USBFADDR(1) | USB Device Functional Address | Go |
| 0x001 | USBPOWER(1) (2) | USB Power | Go |
| 0x002 | USBTXIS(1) (2) | USB Transmit Interrupt Status | Go |
| 0x004 | USBRXIS(1) (2) | USB Receive Interrupt Status | Go |
| 0x006 | USBTXIE(1) (2) | USB Transmit Interrupt Enable | Go |
| 0x008 | USBRXIE(1) (2) | USB Receive Interrupt Enable | Go |
| 0x00A | USBIS(1) (2) | USB General Interrupt Status | Go |
| 0x00B | USBIE(1) (2) | USB Interrupt Enable | Go |
| 0x00C | USBFRAME(1) (2) | USB Frame Value | Go |
| 0x00E | USBEPIDX(1) (2) | USB Endpoint Index | Go |
| 0x00F | USBTEST(1) (2) | USB Test Mode | Go |
| 0x020 | USBFIFO0(1) (2) | USB FIFO Endpoint 0 | Go |
| 0x024 | USBFIFO1(1) (2) | USB FIFO Endpoint 1 | Go |
| 0x028 | USBFIFO2(1) (2) | USB FIFO Endpoint 2 | Go |
| 0x02C | USBFIFO3(1) (2) | USB FIFO Endpoint 3 | Go |
| 0x060 | USBDEVCTL(2) | USB Device Control | Go |
| 0x062 | USBTXFIFOSZ(1) (2) | USB Transmit Dynamic FIFO Sizing | Go |
| 0x063 | USBRXFIFOSZ(1) (2) | USB Receive Dynamic FIFO Sizing | Go |
| 0x064 | USBTXFIFOADD(1) (2) | USB Transmit FIFO Start Address | Go |
| 0x066 | USBRXFIFOADD(1) (2) | USB Receive FIFO Start Address | Go |
| 0x07A | USBCONTIM(1) (2) | USB Connect Timing | Go |
| 0x07D | USBFSEOF(1) (2) | USB Full-Speed Last Transaction to End of Frame Timing | Go |
| 0x07E | USBLSEOF(1) (2) | USB Low-Speed Last Transaction to End of Frame Timing | Go |



Copyright © 2024 Texas Instruments Incorporated

**Table 23-4. Universal Serial Bus (USB) Controller Register Map (continued)**

| Offset | Acronym | Register Name | Section |
|---|---|---|---|
| 0x080 | USBTXFUNCADDR0[2] | USB Transmit Functional Address Endpoint 0 | Go |
| 0x082 | USBTXHUBADDR0[2] | USB Transmit Hub Address Endpoint 0 | Go |
| 0x083 | USBTXHUBPORT0[2] | USB Transmit Hub Port Endpoint 0 | Go |
| 0x088 | USBTXFUNCADDR1[2] | USB Transmit Functional Address Endpoint 1 | Go |
| 0x08A | USBTXHUBADDR1[2] | USB Transmit Hub Address Endpoint 1 | Go |
| 0x08B | USBTXHUBPORT1[2] | USB Transmit Hub Port Endpoint 1 | Go |
| 0x08C | USBRXFUNCADDR1[2] | USB Receive Functional Address Endpoint 1 | Go |
| 0x08E | USBRXHUBADDR1[2] | USB Receive Hub Address Endpoint 1 | Go |
| 0x08F | USBRXHUBPORT1[2] | USB Receive Hub Port Endpoint 1 | Go |
| 0x090 | USBTXFUNCADDR2[2] | USB Transmit Functional Address Endpoint 2 | Go |
| 0x092 | USBTXHUBADDR2[2] | USB Transmit Hub Address Endpoint 2 | Go |
| 0x093 | USBTXHUBPORT2[2] | USB Transmit Hub Port Endpoint 2 | Go |
| 0x094 | USBRXFUNCADDR2[2] | USB Receive Functional Address Endpoint 2 | Go |
| 0x096 | USBRXHUBADDR2[2] | USB Receive Hub Address Endpoint 2 | Go |
| 0x097 | USBRXHUBPORT2[2] | USB Receive Hub Port Endpoint 2 | Go |
| 0x098 | USBTXFUNCADDR3[2] | USB Transmit Functional Address Endpoint 3 | Go |
| 0x09A | USBTXHUBADDR3[2] | USB Transmit Hub Address Endpoint 3 | Go |
| 0x09B | USBTXHUBPORT3[2] | USB Transmit Hub Port Endpoint 3 | Go |
| 0x09C | USBRXFUNCADDR3[2] | USB Receive Functional Address Endpoint 3 | Go |
| 0x09E | USBRXHUBADDR3[2] | USB Receive Hub Address Endpoint 3 | Go |
| 0x09F | USBRXHUBPORT3[2] | USB Receive Hub Port Endpoint 3 | Go |
| 0xA0 | USBTXFUNCADDR4[2] | USB Transmit Functional Address Endpoint 4 | Go |
| 0xA2 | USBTXHUBADDR4[2] | USB Transmit Hub Address Endpoint 4 | Go |
| 0xA3 | USBTXHUBPORT4[2] | USB Transmit Hub Port Endpoint 4 | Go |
| 0xA4 | USBRXFUNCADDR4[2] | USB Receive Functional Address Endpoint 4 | Go |
| 0xA6 | USBRXHUBADDR4[2] | USB Receive Hub Address Endpoint 4 | Go |
| 0xA7 | USBRXHUBPORT4[2] | USB Receive Hub Port Endpoint 4 | Go |
| 0xA8 | USBTXFUNCADDR5[2] | USB Transmit Functional Address Endpoint 5 | Go |
| 0xAA | USBTXHUBADDR5[2] | USB Transmit Hub Address Endpoint 5 | Go |
| 0xAB | USBTXHUBPORT5[2] | USB Transmit Hub Port Endpoint 5 | Go |
| 0xAC | USBRXFUNCADDR4[2] | USB Receive Functional Address Endpoint 5 | Go |
| 0xAE | USBRXHUBADDR5[2] | USB Receive Hub Address Endpoint 5 | Go |
| 0xAF | USBRXHUBPORT5[2] | USB Receive Hub Port Endpoint 5 | Go |
| 0xB0 | USBTXFUNCADDR6[2] | USB Transmit Functional Address Endpoint 6 | Go |
| 0xB2 | USBTXHUBADDR6[2] | USB Transmit Hub Address Endpoint 6 | Go |
| 0xB3 | USBTXHUBPORT6[2] | USB Transmit Hub Port Endpoint 6 | Go |
| 0xB4 | USBRXFUNCADDR6[2] | USB Receive Functional Address Endpoint 6 | Go |
| 0xB6 | USBRXHUBADDR6[2] | USB Receive Hub Address Endpoint 6 | Go |
| 0xB7 | USBRXHUBPORT6[2] | USB Receive Hub Port Endpoint 6 | Go |
| 0xB8 | USBTXFUNCADDR7[2] | USB Transmit Functional Address Endpoint 7 | Go |
| 0xBA | USBTXHUBADDR7[2] | USB Transmit Hub Address Endpoint 7 | Go |
| 0xBB | USBTXHUBPORT7[2] | USB Transmit Hub Port Endpoint 7 | Go |
| 0xBC | USBRXFUNCADDR7[2] | USB Receive Functional Address Endpoint 7 | Go |
| 0xBE | USBRXHUBADDR7[2] | USB Receive Hub Address Endpoint 7 | Go |
| 0xBF | USBRXHUBPORT7[2] | USB Receive Hub Port Endpoint 7 | Go |



Copyright © 2024 Texas Instruments Incorporated

**Table 23-4. Universal Serial Bus (USB) Controller Register Map (continued)**

| Offset | Acronym | Register Name | Section |
|---|---|---|---|
| 0x0C0 | USBTXFUNCADDR8[2] | USB Transmit Functional Address Endpoint 8 | Go |
| 0x0C2 | USBTXHUBADDR8[2] | USB Transmit Hub Address Endpoint 8 | Go |
| 0x0C3 | USBTXHUBPORT8[2] | USB Transmit Hub Port Endpoint 8 | Go |
| 0x0C4 | USBRXFUNCADDR8[2] | USB Receive Functional Address Endpoint 8 | Go |
| 0x0C6 | USBRXHUBADDR8[2] | USB Receive Hub Address Endpoint 8 | Go |
| 0x0C7 | USBRXHUBPORT8[2] | USB Receive Hub Port Endpoint 8 | Go |
| 0xC8 | USBTXFUNCADDR9[2] | USB Transmit Functional Address Endpoint 9 | Go |
| 0x0CA | USBTXHUBADDR9[2] | USB Transmit Hub Address Endpoint 9 | Go |
| 0x0CB | USBTXHUBPORT9[2] | USB Transmit Hub Port Endpoint 9 | Go |
| 0x0CC | USBRXFUNCADDR9[2] | USB Receive Functional Address Endpoint 9 | Go |
| 0x0CE | USBRXHUBADDR9[2] | USB Receive Hub Address Endpoint 9 | Go |
| 0x0CF | USBRXHUBPORT9[2] | USB Receive Hub Port Endpoint 9 | Go |
| 0x0D0 | USBTXFUNCADDR10[2] | USB Transmit Functional Address Endpoint 10 | Go |
| 0x0D2 | USBTXHUBADDR10[2] | USB Transmit Hub Address Endpoint 10 | Go |
| 0x0D3 | USBTXHUBPORT10[2] | USB Transmit Hub Port Endpoint 10 | Go |
| 0x0D4 | USBRXFUNCADDR10[2] | USB Receive Functional Address Endpoint 10 | Go |
| 0x0D6 | USBRXHUBADDR10[2] | USB Receive Hub Address Endpoint 10 | Go |
| 0x0D7 | USBRXHUBPORT10[2] | USB Receive Hub Port Endpoint 10 | Go |
| 0x0D8 | USBTXFUNCADDR11[2] | USB Transmit Functional Address Endpoint 11 | Go |
| 0x0DA | USBTXHUBADDR11[2] | USB Transmit Hub Address Endpoint 11 | Go |
| 0x0DB | USBTXHUBPORT11[2] | USB Transmit Hub Port Endpoint 11 | Go |
| 0x0DC | USBRXFUNCADDR11[2] | USB Receive Functional Address Endpoint 11 | Go |
| 0x0DE | USBRXHUBADDR11[2] | USB Receive Hub Address Endpoint 11 | Go |
| 0x0DF | USBRXHUBPORT11[2] | USB Receive Hub Port Endpoint 11 | Go |
| 0x0E0 | USBTXFUNCADDR12[2] | USB Transmit Functional Address Endpoint 12 | Go |
| 0x0E2 | USBTXHUBADDR12[2] | USB Transmit Hub Address Endpoint 12 | Go |
| 0x0E3 | USBTXHUBPORT12[2] | USB Transmit Hub Port Endpoint 12 | Go |
| 0x0E4 | USBRXFUNCADDR12[2] | USB Receive Functional Address Endpoint 12 | Go |
| 0x0E6 | USBRXHUBADDR12[2] | USB Receive Hub Address Endpoint 12 | Go |
| 0x0E7 | USBRXHUBPORT12[2] | USB Receive Hub Port Endpoint 12 | Go |
| 0x0E8 | USBTXFUNCADDR13[2] | USB Transmit Functional Address Endpoint 13 | Go |
| 0x0EA | USBTXHUBADDR13[2] | USB Transmit Hub Address Endpoint 13 | Go |
| 0x0EB | USBTXHUBPORT13[2] | USB Transmit Hub Port Endpoint 13 | Go |
| 0x0EC | USBRXFUNCADDR13[2] | USB Receive Functional Address Endpoint 13 | Go |
| 0x0EE | USBRXHUBADDR13[2] | USB Receive Hub Address Endpoint 13 | Go |
| 0x0EF | USBRXHUBPORT13[2] | USB Receive Hub Port Endpoint 13 | Go |
| 0x0F0 | USBTXFUNCADDR14[2] | USB Transmit Functional Address Endpoint 14 | Go |
| 0x0F2 | USBTXHUBADDR14[2] | USB Transmit Hub Address Endpoint 14 | Go |
| 0x0F3 | USBTXHUBPORT14[2] | USB Transmit Hub Port Endpoint 14 | Go |
| 0x0F4 | USBRXFUNCADDR14[2] | USB Receive Functional Address Endpoint 14 | Go |
| 0x0F6 | USBRXHUBADDR14[2] | USB Receive Hub Address Endpoint 14 | Go |
| 0x0F7 | USBRXHUBPORT14[2] | USB Receive Hub Port Endpoint 14 | Go |
| 0x0F8 | USBTXFUNCADDR15[2] | USB Transmit Functional Address Endpoint 15 | Go |
| 0x0FA | USBTXHUBADDR15[2] | USB Transmit Hub Address Endpoint 15 | Go |
| 0x0FB | USBTXHUBPORT15[2] | USB Transmit Hub Port Endpoint 15 | Go |



Copyright © 2024 Texas Instruments Incorporated

**Table 23-4. Universal Serial Bus (USB) Controller Register Map (continued)**

| Offset | Acronym | Register Name | Section |
|---|---|---|---|
| 0x0FC | USBRXFUNCADDR15[2] | USB Receive Functional Address Endpoint 15 | Go |
| 0x0FE | USBRXHUBADDR15[2] | USB Receive Hub Address Endpoint 15 | Go |
| 0x0FF | USBRXHUBPORT15[2] | USB Receive Hub Port Endpoint 15 | Go |
| 0x102 | USBCSRL0[1] [2] | USB Control and Status Endpoint 0 Low | Go |
| 0x103 | USBCSRH0[1] [2] | USB Control and Status Endpoint 0 High | Go |
| 0x108 | USBCOUNT0[1] [2] | USB Receive Byte Count Endpoint 0 | Go |
| 0x10A | USBTYPE0[2] | USB Type Endpoint 0 | Go |
| 0x10B | USBNAKLMT[2] | USB NAK Limit | Go |
| 0x110 | USBTXMAXP1[1] [2] | USB Maximum Transmit Data Endpoint 1 | Go |
| 0x112 | USBTXCSRL1[1] [2] | USB Transmit Control and Status Endpoint 1 Low | Go |
| 0x113 | USBTXCSRH1[1] [2] | USB Transmit Control and Status Endpoint 1 High | Go |
| 0x114 | USBRXMAXP1[1] [2] | USB Maximum Receive Data Endpoint 1 | Go |
| 0x116 | USBRXCSRL1[1] [2] | USB Receive Control and Status Endpoint 1 Low | Go |
| 0x117 | USBRXCSRH1[1] [2] | USB Receive Control and Status Endpoint 1 High | Go |
| 0x118 | USBRXCOUNT1[1] [2] | USB Receive Byte Count Endpoint 1 | Go |
| 0x11A | USBTXTYPE1[2] | USB Host Transmit Configure Type Endpoint 1 | Go |
| 0x11B | USBTXINTERVAL1[2] | USB Host Transmit Interval Endpoint 1 | Go |
| 0x11C | USBRXTYPE1[2] | USB Host Configure Receive Type Endpoint 1 | Go |
| 0x11D | USBRXINTERVAL1[2] | USB Host Receive Polling Interval Endpoint 1 | Go |
| 0x120 | USBTXMAXP2[1] [2] | USB Maximum Transmit Data Endpoint 2 | Go |
| 0x122 | USBTXCSRL2[1] [2] | USB Transmit Control and Status Endpoint 2 Low | Go |
| 0x123 | USBTXCSRH2[1] [2] | USB Transmit Control and Status Endpoint 2 High | Go |
| 0x124 | USBRXMAXP2[1] [2] | USB Maximum Receive Data Endpoint 2 | Go |
| 0x126 | USBRXCSRL2[1] [2] | USB Receive Control and Status Endpoint 2 Low | Go |
| 0x127 | USBRXCSRH2[1] [2] | USB Receive Control and Status Endpoint 2 High | Go |
| 0x128 | USBRXCOUNT2[1] [2] | USB Receive Byte Count Endpoint 2 | Go |
| 0x12A | USBTXTYPE2[2] | USB Host Transmit Configure Type Endpoint 2 | Go |
| 0x12B | USBTXINTERVAL2[2] | USB Host Transmit Interval Endpoint 2 | Go |
| 0x12C | USBRXTYPE2[2] | USB Host Configure Receive Type Endpoint 2 | Go |
| 0x12D | USBRXINTERVAL2[2] | USB Host Receive Polling Interval Endpoint 2 | Go |
| 0x130 | USBTXMAXP3[1] [2] | USB Maximum Transmit Data Endpoint 3 | Go |
| 0x132 | USBTXCSRL3[1] [2] | USB Transmit Control and Status Endpoint 3 Low | Go |
| 0x133 | USBTXCSRH3[1] [2] | USB Transmit Control and Status Endpoint 3 High | Go |
| 0x134 | USBRXMAXP3[1] [2] | USB Maximum Receive Data Endpoint 3 | Go |
| 0x136 | USBRXCSRL3[1] [2] | USB Receive Control and Status Endpoint 3 Low | Go |
| 0x137 | USBRXCSRH3[1] [2] | USB Receive Control and Status Endpoint 3 High | Go |
| 0x138 | USBRXCOUNT3[1] [2] | USB Receive Byte Count Endpoint 3 | Go |
| 0x13A | USBTXTYPE3[2] | USB Host Transmit Configure Type Endpoint 3 | Go |
| 0x13B | USBTXINTERVAL3[2] | USB Host Transmit Interval Endpoint 3 | Go |
| 0x13C | USBRXTYPE3[2] | USB Host Configure Receive Type Endpoint 3 | Go |
| 0x13D | USBRXINTERVAL3[2] | USB Host Receive Polling Interval Endpoint 3 | Go |
| 0x140 | USBTXMAXP4[1] [2] | USB Maximum Transmit Data Endpoint 4 | Go |
| 0x142 | USBTXCSRL4[1] [2] | USB Transmit Control and Status Endpoint 4 Low | Go |
| 0x143 | USBTXCSRH4[1] [2] | USB Transmit Control and Status Endpoint 4 High | Go |
| 0x144 | USBRXMAXP4[1] [2] | USB Maximum Receive Data Endpoint 4 | Go |



Copyright © 2024 Texas Instruments Incorporated

**Table 23-4. Universal Serial Bus (USB) Controller Register Map (continued)**

| Offset | Acronym | Register Name | Section |
|---|---|---|---|
| 0x146 | USBRXCSRL4[1] [2] | USB Receive Control and Status Endpoint 4 Low | Go |
| 0x147 | USBRXCSRH4[1] [2] | USB Receive Control and Status Endpoint 4 High | Go |
| 0x148 | USBRXCOUNT4[1] [2] | USB Receive Byte Count Endpoint 4 | Go |
| 0x14A | USBTXTYPE4[2] | USB Host Transmit Configure Type Endpoint 4 | Go |
| 0x14B | USBTXINTERVAL4[2] | USB Host Transmit Interval Endpoint 4 | Go |
| 0x14C | USBRXTYPE4[2] | USB Host Configure Receive Type Endpoint 4 | Go |
| 0x14D | USBRXINTERVAL4[2] | USB Host Receive Polling Interval Endpoint 4 | Go |
| 0x150 | USBTXMAXP5[1] [2] | USB Maximum Transmit Data Endpoint 5 | Go |
| 0x152 | USBTXCSRL5[1] [2] | USB Transmit Control and Status Endpoint 5 Low | Go |
| 0x153 | USBTXCSRH5[1] [2] | USB Transmit Control and Status Endpoint 5 High | Go |
| 0x154 | USBRXMAXP5[1] [2] | USB Maximum Receive Data Endpoint 5 | Go |
| 0x156 | USBRXCSRL5[1] [2] | USB Receive Control and Status Endpoint 5 Low | Go |
| 0x157 | USBRXCSRH5[1] [2] | USB Receive Control and Status Endpoint 5 High | Go |
| 0x158 | USBRXCOUNT5[1] [2] | USB Receive Byte Count Endpoint 5 | Go |
| 0x15A | USBTXTYPE5[2] | USB Host Transmit Configure Type Endpoint 5 | Go |
| 0x15B | USBTXINTERVAL5[2] | USB Host Transmit Interval Endpoint 5 | Go |
| 0x15C | USBRXTYPE5[2] | USB Host Configure Receive Type Endpoint 5 | Go |
| 0x15D | USBRXINTERVAL5[2] | USB Host Receive Polling Interval Endpoint 5 | Go |
| 0x160 | USBTXMAXP6[1] [2] | USB Maximum Transmit Data Endpoint 6 | Go |
| 0x162 | USBTXCSRL6[1] [2] | USB Transmit Control and Status Endpoint 6 Low | Go |
| 0x163 | USBTXCSRH6[1] [2] | USB Transmit Control and Status Endpoint 6 High | Go |
| 0x164 | USBRXMAXP6[1] [2] | USB Maximum Receive Data Endpoint 6 | Go |
| 0x166 | USBRXCSRL6[1] [2] | USB Receive Control and Status Endpoint 6 Low | Go |
| 0x167 | USBRXCSRH6[1] [2] | USB Receive Control and Status Endpoint 6 High | Go |
| 0x168 | USBRXCOUNT6[1] [2] | USB Receive Byte Count Endpoint 6 | Go |
| 0x16A | USBTXTYPE6[2] | USB Host Transmit Configure Type Endpoint 6 | Go |
| 0x16B | USBTXINTERVAL6[2] | USB Host Transmit Interval Endpoint 6 | Go |
| 0x16C | USBRXTYPE6[2] | USB Host Configure Receive Type Endpoint 6 | Go |
| 0x16D | USBRXINTERVAL6[2] | USB Host Receive Polling Interval Endpoint 6 | Go |
| 0x170 | USBTXMAXP7[1] [2] | USB Maximum Transmit Data Endpoint 7 | Go |
| 0x172 | USBTXCSRL7[1] [2] | USB Transmit Control and Status Endpoint 7 Low | Go |
| 0x173 | USBTXCSRH7[1] [2] | USB Transmit Control and Status Endpoint 7 High | Go |
| 0x174 | USBRXMAXP7[1] [2] | USB Maximum Receive Data Endpoint 7 | Go |
| 0x176 | USBRXCSRL7[1] [2] | USB Receive Control and Status Endpoint 7 Low | Go |
| 0x177 | USBRXCSRH7[1] [2] | USB Receive Control and Status Endpoint 7 High | Go |
| 0x178 | USBRXCOUNT7[1] [2] | USB Receive Byte Count Endpoint 7 | Go |
| 0x17A | USBTXTYPE7[2] | USB Host Transmit Configure Type Endpoint 7 | Go |
| 0x17B | USBTXINTERVAL7[2] | USB Host Transmit Interval Endpoint 7 | Go |
| 0x17C | USBRXTYPE7[2] | USB Host Configure Receive Type Endpoint 7 | Go |
| 0x17D | USBRXINTERVAL7[2] | USB Host Receive Polling Interval Endpoint 7 | Go |
| 0x180 | USBTXMAXP8[1] [2] | USB Maximum Transmit Data Endpoint 8 | Go |
| 0x182 | USBTXCSRL8[1] [2] | USB Transmit Control and Status Endpoint 8 Low | Go |
| 0x183 | USBTXCSRH8[1] [2] | USB Transmit Control and Status Endpoint 8 High | Go |
| 0x184 | USBRXMAXP8[1] [2] | USB Maximum Receive Data Endpoint 8 | Go |
| 0x186 | USBRXCSRL8[1] [2] | USB Receive Control and Status Endpoint 8 Low | Go |



Copyright © 2024 Texas Instruments Incorporated

**Table 23-4. Universal Serial Bus (USB) Controller Register Map (continued)**

| Offset | Acronym | Register Name | Section |
|---|---|---|---|
| 0x187 | USBRXCSRH8[1] [2] | USB Receive Control and Status Endpoint 8 High | Go |
| 0x188 | USBRXCOUNT8[1] [2] | USB Receive Byte Count Endpoint 8 | Go |
| 0x18A | USBTXTYPE8[2] | USB Host Transmit Configure Type Endpoint 8 | Go |
| 0x18B | USBTXINTERVAL8[2] | USB Host Transmit Interval Endpoint 8 | Go |
| 0x18C | USBRXTYPE8[2] | USB Host Configure Receive Type Endpoint 8 | Go |
| 0x18D | USBRXINTERVAL8[2] | USB Host Receive Polling Interval Endpoint 8 | Go |
| 0x190 | USBTXMAXP9[1] [2] | USB Maximum Transmit Data Endpoint 9 | Go |
| 0x192 | USBTXCSRL9[1] [2] | USB Transmit Control and Status Endpoint 9 Low | Go |
| 0x193 | USBTXCSRH9[1] [2] | USB Transmit Control and Status Endpoint 9 High | Go |
| 0x194 | USBRXMAXP9[1] [2] | USB Maximum Receive Data Endpoint 9 | Go |
| 0x196 | USBRXCSRL9[1] [2] | USB Receive Control and Status Endpoint 9 Low | Go |
| 0x197 | USBRXCSRH9[1] [2] | USB Receive Control and Status Endpoint 9 High | Go |
| 0x198 | USBRXCOUNT9[1] [2] | USB Receive Byte Count Endpoint 9 | Go |
| 0x19A | USBTXTYPE9[2] | USB Host Transmit Configure Type Endpoint 9 | Go |
| 0x19B | USBTXINTERVAL9[2] | USB Host Transmit Interval Endpoint 9 | Go |
| 0x19C | USBRXTYPE9[2] | USB Host Configure Receive Type Endpoint 9 | Go |
| 0x19D | USBRXINTERVAL9[2] | USB Host Receive Polling Interval Endpoint 9 | Go |
| 0x1A0 | USBTXMAXP10[1] [2] | USB Maximum Transmit Data Endpoint 10 | Go |
| 0x1A2 | USBTXCSRL10[1] [2] | USB Transmit Control and Status Endpoint 10 Low | Go |
| 0x1A3 | USBTXCSRH10[1] [2] | USB Transmit Control and Status Endpoint 10 High | Go |
| 0x1A4 | USBRXMAXP10[1] [2] | USB Maximum Receive Data Endpoint 10 | Go |
| 0x1A6 | USBRXCSRL10[1] [2] | USB Receive Control and Status Endpoint 10 Low | Go |
| 0x1A7 | USBRXCSRH10[1] [2] | USB Receive Control and Status Endpoint 10 High | Go |
| 0x1A8 | USBRXCOUNT10[1] [2] | USB Receive Byte Count Endpoint 10 | Go |
| 0x1AA | USBTXTYPE10[2] | USB Host Transmit Configure Type Endpoint 10 | Go |
| 0x1AB | USBTXINTERVAL10[2] | USB Host Transmit Interval Endpoint 10 | Go |
| 0x1AC | USBRXTYPE10[2] | USB Host Configure Receive Type Endpoint 10 | Go |
| 0x1AD | USBRXINTERVAL10[2] | USB Host Receive Polling Interval Endpoint 10 | Go |
| 0x1B0 | USBTXMAXP11[1] [2] | USB Maximum Transmit Data Endpoint 11 | Go |
| 0x1B2 | USBTXCSRL11[1] [2] | USB Transmit Control and Status Endpoint 11 Low | Go |
| 0x1B3 | USBTXCSRH11[1] [2] | USB Transmit Control and Status Endpoint 11 High | Go |
| 0x1B4 | USBRXMAXP11[1] [2] | USB Maximum Receive Data Endpoint 11 | Go |
| 0x1B6 | USBRXCSRL11[1] [2] | USB Receive Control and Status Endpoint 11 Low | Go |
| 0x1B7 | USBRXCSRH11[1] [2] | USB Receive Control and Status Endpoint 11 High | Go |
| 0x1B8 | USBRXCOUNT11[1] [2] | USB Receive Byte Count Endpoint 11 | Go |
| 0x1BA | USBTXTYPE11[2] | USB Host Transmit Configure Type Endpoint 11 | Go |
| 0x1BB | USBTXINTERVAL11[2] | USB Host Transmit Interval Endpoint 11 | Go |
| 0x1BC | USBRXTYPE11[2] | USB Host Configure Receive Type Endpoint 11 | Go |
| 0x1BD | USBRXINTERVAL11[2] | USB Host Receive Polling Interval Endpoint 11 | Go |
| 0x1C0 | USBTXMAXP12[1] [2] | USB Maximum Transmit Data Endpoint 12 | Go |
| 0x1C2 | USBTXCSRL12[1] [2] | USB Transmit Control and Status Endpoint 12 Low | Go |
| 0x1C3 | USBTXCSRH12[1] [2] | USB Transmit Control and Status Endpoint 12 High | Go |
| 0x1C4 | USBRXMAXP12[1] [2] | USB Maximum Receive Data Endpoint 12 | Go |
| 0x1C6 | USBRXCSRL12[1] [2] | USB Receive Control and Status Endpoint 12 Low | Go |
| 0x1C7 | USBRXCSRH12[1] [2] | USB Receive Control and Status Endpoint 12 High | Go |



Copyright © 2024 Texas Instruments Incorporated

**Table 23-4. Universal Serial Bus (USB) Controller Register Map (continued)**

| Offset | Acronym | Register Name | Section |
|---|---|---|---|
| 0x1C8 | USBRXCOUNT12(1) (2) | USB Receive Byte Count Endpoint 12 | Go |
| 0x1CA | USBTXTYPE12(2) | USB Host Transmit Configure Type Endpoint 12 | Go |
| 0x1CB | USBTXINTERVAL12(2) | USB Host Transmit Interval Endpoint 12 | Go |
| 0x1CC | USBRXTYPE12(2) | USB Host Configure Receive Type Endpoint 12 | Go |
| 0x1CD | USBRXINTERVAL12(2) | USB Host Receive Polling Interval Endpoint 12 | Go |
| 0x1D0 | USBTXMAXP13(1) (2) | USB Maximum Transmit Data Endpoint 13 | Go |
| 0x1D2 | USBTXCSRL13(1) (2) | USB Transmit Control and Status Endpoint 13 Low | Go |
| 0x1D3 | USBTXCSRH13(1) (2) | USB Transmit Control and Status Endpoint 13 High | Go |
| 0x1D4 | USBRXMAXP13(1) (2) | USB Maximum Receive Data Endpoint 13 | Go |
| 0x1D6 | USBRXCSRL13(1) (2) | USB Receive Control and Status Endpoint 13 Low | Go |
| 0x1D7 | USBRXCSRH13(1) (2) | USB Receive Control and Status Endpoint 13 High | Go |
| 0x1D8 | USBRXCOUNT13(1) (2) | USB Receive Byte Count Endpoint 13 | Go |
| 0x1DA | USBTXTYPE13(2) | USB Host Transmit Configure Type Endpoint 13 | Go |
| 0x1DB | USBTXINTERVAL13(2) | USB Host Transmit Interval Endpoint 13 | Go |
| 0x1DC | USBRXTYPE13(2) | USB Host Configure Receive Type Endpoint 13 | Go |
| 0x1DD | USBRXINTERVAL13(2) | USB Host Receive Polling Interval Endpoint 13 | Go |
| 0x1E0 | USBTXMAXP14(1) (2) | USB Maximum Transmit Data Endpoint 14 | Go |
| 0x1E2 | USBTXCSRL14(1) (2) | USB Transmit Control and Status Endpoint 14 Low | Go |
| 0x1E3 | USBTXCSRH14(1) (2) | USB Transmit Control and Status Endpoint 14 High | Go |
| 0x1E4 | USBRXMAXP14(1) (2) | USB Maximum Receive Data Endpoint 14 | Go |
| 0x1E6 | USBRXCSRL14(1) (2) | USB Receive Control and Status Endpoint 14 Low | Go |
| 0x1E7 | USBRXCSRH14(1) (2) | USB Receive Control and Status Endpoint 14 High | Go |
| 0x1E8 | USBRXCOUNT14(1) (2) | USB Receive Byte Count Endpoint 14 | Go |
| 0x1EA | USBTXTYPE14(2) | USB Host Transmit Configure Type Endpoint 14 | Go |
| 0x1EB | USBTXINTERVAL14(2) | USB Host Transmit Interval Endpoint 14 | Go |
| 0x1EC | USBRXTYPE14(2) | USB Host Configure Receive Type Endpoint 14 | Go |
| 0x1ED | USBRXINTERVAL14(2) | USB Host Receive Polling Interval Endpoint 14 | Go |
| 0x1F0 | USBTXMAXP15(1) (2) | USB Maximum Transmit Data Endpoint 15 | Go |
| 0x1F2 | USBTXCSRL15(1) (2) | USB Transmit Control and Status Endpoint 15 Low | Go |
| 0x1F3 | USBTXCSRH15(1) (2) | USB Transmit Control and Status Endpoint 15 High | Go |
| 0x1F4 | USBRXMAXP15(1) (2) | USB Maximum Receive Data Endpoint 15 | Go |
| 0x1F6 | USBRXCSRL15(1) (2) | USB Receive Control and Status Endpoint 15 Low | Go |
| 0x1F7 | USBRXCSRH15(1) (2) | USB Receive Control and Status Endpoint 15 High | Go |
| 0x1F8 | USBRXCOUNT15(1) (2) | USB Receive Byte Count Endpoint 15 | Go |
| 0x1FA | USBTXTYPE15(2) | USB Host Transmit Configure Type Endpoint 15 | Go |
| 0x1FB | USBTXINTERVAL15(2) | USB Host Transmit Interval Endpoint 15 | Go |
| 0x1FC | USBRXTYPE15(2) | USB Host Configure Receive Type Endpoint 15 | Go |
| 0x1FD | USBRXINTERVAL15(2) | USB Host Receive Polling Interval Endpoint 15 | Go |
| 0x304 | USBRQPKTCOUNT1(2) | USB Request Packet Count in Block Transfer Endpoint 1 | Go |
| 0x308 | USBRQPKTCOUNT2(2) | USB Request Packet Count in Block Transfer Endpoint 2 | Go |
| 0x30C | USBRQPKTCOUNT3(2) | USB Request Packet Count in Block Transfer Endpoint 3 | Go |
| 0x310 | USBRQPKTCOUNT4(2) | USB Request Packet Count in Block Transfer Endpoint 4 | Go |
| 0x314 | USBRQPKTCOUNT5(2) | USB Request Packet Count in Block Transfer Endpoint 5 | Go |
| 0x318 | USBRQPKTCOUNT6(2) | USB Request Packet Count in Block Transfer Endpoint 6 | Go |
| 0x31C | USBRQPKTCOUNT7(2) | USB Request Packet Count in Block Transfer Endpoint 7 | Go |



Copyright © 2024 Texas Instruments Incorporated

**Table 23-4. Universal Serial Bus (USB) Controller Register Map (continued)**

| Offset | Acronym | Register Name | Section |
|---|---|---|---|
| 0x320 | USBRQPKTCOUNT8[2] | USB Request Packet Count in Block Transfer Endpoint 8 | Go |
| 0x324 | USBRQPKTCOUNT9[2] | USB Request Packet Count in Block Transfer Endpoint 9 | Go |
| 0x328 | USBRQPKTCOUNT10[2] | USB Request Packet Count in Block Transfer Endpoint 10 | Go |
| 0x32C | USBRQPKTCOUNT11[2] | USB Request Packet Count in Block Transfer Endpoint 11 | Go |
| 0x330 | USBRQPKTCOUNT12[2] | USB Request Packet Count in Block Transfer Endpoint 12 | Go |
| 0x334 | USBRQPKTCOUNT13[2] | USB Request Packet Count in Block Transfer Endpoint 13 | Go |
| 0x338 | USBRQPKTCOUNT14[2] | USB Request Packet Count in Block Transfer Endpoint 14 | Go |
| 0x33C | USBRQPKTCOUNT15[2] | USB Request Packet Count in Block Transfer Endpoint 15 | Go |
| 0x340 | USBRXDPKTBUFDIS[1] [2] | USB Receive Double Packet Buffer Disable | Go |
| 0x342 | USBTXDPKTBUFDIS[1] [2] | USB Transmit Double Packet Buffer Disable | Go |
| 0x400 | USBEPC[1] [2] | USB External Power Control | Go |
| 0x404 | USBEPCRIS[1] [2] | USB External Power Control Raw Interrupt Status | Go |
| 0x408 | USBEPCIM[2] [1] | USB External Power Control Interrupt Mask | Go |
| 0x40C | USBEPCISC[1] [2] | USB External Power Control Interrupt Status and Clear | Go |
| 0x410 | USBDRRIS[1] [2] | USB Device RESUME Raw Interrupt Status | Go |
| 0x414 | USBDRIM[1] [2] | USB Device RESUME Interrupt Mask | Go |
| 0x418 | USBDRISC[1] [2] | USB Device RESUME Interrupt Status and Clear | Go |
| 0x41C | USBGPCS[1] [2] | USB General-Purpose Control and Status | Go |
| 0x450 | USBDMASEL[1] [2] | USB DMA Select | Go |

(1) This register is used in Device mode. Some registers are used for both Host and Device mode and can have different bit definitions depending on the mode.

(2) This register is used in Host mode. Some registers are used for both Host and Device mode and can have different bit definitions depending on the mode. The USB controller is in Device mode upon reset, so the reset values shown for these registers apply to the Device mode definition.



Copyright © 2024 Texas Instruments Incorporated

### 23.6.3 Register Descriptions

#### 23.6.3.1 USB Device Functional Address Register (USBFADDR), offset 0x000

The USB function address 8-bit register (USBFADDR) contains the 7-bit address of the device part of the transaction.

When the USB controller is being used in device mode (the HOST bit in the USBDEVCTL register is clear), this register must be written with the address received through a SET_ADDRESS command, which is then used for decoding the function address in subsequent token packets.

**Mode(s):** Device

For special considerations when writing this register, see the *Setting the Device Address* in Section 23.2.1.1.4.

USBFADDR is shown in Figure 23-3 and described in Table 23-5.

**Figure 23-3. USB Device Functional Address Register (USBFADDR)**

| 7 | 6 | 0 |
|---|---|---|
| Reserved | FUNCADDR | |
| R-0 | R/W-0 | |

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

**Table 23-5. USB Device Functional Address Register (USBFADDR) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7 | Reserved | 0 | Reserved |
| 6-0 | FUNCADDR | 0-7Fh | Function Address of Device as received through SET_ADDRESS. |



Copyright © 2024 Texas Instruments Incorporated

#### 23.6.3.2 USB Power Management Register (USBPOWER), offset 0x001

The power management 8-bit register (USBPOWER) is used for controlling SUSPEND and RESUME signaling, and some basic operational aspects of the USB controller.

**Mode(s):** Host Device

USBPOWER in Host Mode is shown in Figure 23-4 and described in Table 23-6.

**Figure 23-4. USB Power Management Register (USBPOWER) in Host Mode**

| 7 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|
| Reserved | | RESET | RESUME | SUSPEND | PWRDNPHY |
| R-0 | | R/W-0 | R/W-0 | R/W-1S | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

**Table 23-6. USB Power Management Register (USBPOWER) in Host Mode Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7-4 | Reserved | 0 | Reserved |
| 3 | RESET | | RESET signaling. |
| | | 0 | Ends RESET signaling on the bus. |
| | | 1 | Enables RESET signaling on the bus. |
| 2 | RESUME | | RESUME signaling. The bit can be cleared by software 20 ms after being set. |
| | | 0 | Ends RESUME signaling on the bus. |
| | | 1 | Enables RESUME signaling when the Device is in SUSPEND mode. |
| 1 | SUSPEND | | SUSPEND mode |
| | | 0 | No effect |
| | | 1 | Enables SUSPEND mode. |
| 0 | PWRDNPHY | | Power Down PHY |
| | | 0 | No effect |
| | | 1 | Powers down the internal USB PHY. |



Copyright © 2024 Texas Instruments Incorporated

USBPOWER in Device Mode is shown in Figure 23-5 and described in Table 23-7.

**Figure 23-5. USB Power Management Register (USBPOWER) in Device Mode**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| ISOUPDATE | SOFTCONN | Reserved | | RESET | RESUME | SUSPEND | PWRDNPHY |
| R/W-0 | R/W-0 | R-0 | | R/W-0 | R/W-0 | R-0 | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

**Table 23-7. USB Power Management Register (USBPOWER) in Device Mode Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7 | Reserved | | Reserved |
| 6 | SOFTCONN | | Soft Connect/Disconnect |
| | | 0 | The USB D+/D- lines are tri-stated. |
| | | 1 | The USB D+/D- lines are enabled. |
| 5-4 | Reserved | 0 | Reserved |
| 3 | RESET | | RESET signaling |
| | | 0 | Ends RESET signaling on the bus. |
| | | 1 | Enables RESET signaling on the bus. |
| 2 | RESUME | | RESUME signaling. The bit can be cleared by software 10 ms (a maximum of 15 ms) after being set. |
| | | 0 | Ends RESUME signaling on the bus. |
| | | 1 | Enables RESUME signaling when the Device is in SUSPEND mode. |
| 1 | SUSPEND | | SUSPEND mode. |
| | | 0 | This bit is cleared when software reads the interrupt register or sets the RESUME bit above. |
| | | 1 | The USB controller is in SUSPEND mode. |
| 0 | PWRDNPHY | | Power Down PHY |
| | | 0 | No effect |
| | | 1 | Powers down the internal USB PHY. |



Copyright © 2024 Texas Instruments Incorporated

#### 23.6.3.3 USB Transmit Interrupt Status Register (USBTXIS), offset 0x002

**Note**

Use caution when reading this register. Performing a read may change bit status.

The USB transmit interrupt status 16-bit read-only register (USBTXIS) indicates which interrupts are currently active for control endpoint 0 and the transmit endpoints 1–15. The meaning of the EPn bits in this register is based on the mode of the device. The EP1 through EP15 bits always indicate that the USB controller is sending data; however, in Host mode, the bits refer to OUT endpoints; while in Device mode, the bits refer to IN endpoints.

**Note:**The EP0 bit is special in Host and Device modes and indicates that either a control IN or control OUT endpoint has generated an interrupt. Both the control IN and control OUT endpoints are captured in the EP0 bit of the USBTXIS register.

**Mode(s):** Host Device

USBTXIS is shown in Figure 23-6 and described in Table 23-8.

**Figure 23-6. USB Transmit Interrupt Status Register (USBTXIS)**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| EP15 | EP14 | EP13 | EP12 | EP11 | EP10 | EP9 | EP8 |
| R-0 | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EP7 | EP6 | EP5 | EP4 | EP3 | EP2 | EP1 | EP0 |
| R-0 | | | | | | | |

**Table 23-8. USB Transmit Interrupt Status Register (USBTXIS) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 15 | EP15 | | TX Endpoint 15 Interrupt |
| | | 0 | No interrupt |
| | | 1 | The Endpoint 15 transmit interrupt is asserted. |
| 14 | EP14 | | TX Endpoint 14 Interrupt |
| | | 0 | No interrupt |
| | | 1 | The Endpoint 14 transmit interrupt is asserted. |
| 13 | EP13 | | TX Endpoint 13 Interrupt |
| | | 0 | No interrupt |
| | | 1 | The Endpoint 13 transmit interrupt is asserted. |
| 12 | EP12 | | TX Endpoint 12 Interrupt |
| | | 0 | No interrupt |
| | | 1 | The Endpoint 12 transmit interrupt is asserted. |
| 11 | EP11 | | TX Endpoint 11 Interrupt |
| | | 0 | No interrupt |
| | | 1 | The Endpoint 11 transmit interrupt is asserted. |
| 10 | EP10 | | TX Endpoint 10 Interrupt |
| | | 0 | No interrupt |
| | | 1 | The Endpoint 10 transmit interrupt is asserted. |



Copyright © 2024 Texas Instruments Incorporated

**Table 23-8. USB Transmit Interrupt Status Register (USBTXIS) Field Descriptions (continued)**

| Bit | Field | Value | Description |
|---|---|---|---|
| 9 | EP9 | | TX Endpoint 9 Interrupt |
| | | 0 | No interrupt |
| | | 1 | The Endpoint 9 transmit interrupt is asserted. |
| 8 | EP8 | | TX Endpoint 8 Interrupt |
| | | 0 | No interrupt |
| | | 1 | The Endpoint 8 transmit interrupt is asserted. |
| 7 | EP7 | | TX Endpoint 7 Interrupt |
| | | 0 | No interrupt |
| | | 1 | The Endpoint 7 transmit interrupt is asserted. |
| 6 | EP6 | | TX Endpoint 6 Interrupt |
| | | 0 | No interrupt |
| | | 1 | The Endpoint 6 transmit interrupt is asserted. |
| 5 | EP5 | | TX Endpoint 5 Interrupt |
| | | 0 | No interrupt |
| | | 1 | The Endpoint 5 transmit interrupt is asserted. |
| 4 | EP4 | | TX Endpoint 4 Interrupt |
| | | 0 | No interrupt |
| | | 1 | The Endpoint 4 transmit interrupt is asserted. |
| 3 | EP3 | | TX Endpoint 3 Interrupt |
| | | 0 | No interrupt |
| | | 1 | The Endpoint 3 transmit interrupt is asserted. |
| 2 | EP2 | | TX Endpoint 2 Interrupt |
| | | 0 | No interrupt |
| | | 1 | The Endpoint 2 transmit interrupt is asserted. |
| 1 | EP1 | | TX Endpoint 1 Interrupt |
| | | 0 | No interrupt |
| | | 1 | The Endpoint 1 transmit interrupt is asserted. |
| 0 | EP0 | | TX and RX Endpoint 0 Interrupt |
| | | 0 | No interrupt |
| | | 1 | The Endpoint 0 transmit and receive interrupt is asserted. |



Copyright © 2024 Texas Instruments Incorporated

#### 23.6.3.4 USB Receive Interrupt Status Register (USBRXIS), offset 0x004

**Note**

Use caution when reading this register. Performing a read may change bit status.

The USB receive interrupt status 16-bit read-only register (USBRXIS) indicates which interrupts are currently active for receive endpoints 1–15.

**Note:** The USBRXIS register does not have a bit for EP0. See the USBTXIS register for EP0 use.

**Note:** Bits relating to endpoints that have not been configured always return 0. All active interrupts are cleared when this register is read.

**Mode(s):** Host Device

USBRXIS is shown in Figure 23-7 and described in Table 23-9.

**Figure 23-7. USB Transmit Interrupt Status Register (USBRXIS)**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| EP15 | EP14 | EP13 | EP12 | EP11 | EP10 | EP9 | EP8 |
| R-0 | | | | | | | |
| EP7 | EP6 | EP5 | EP4 | EP3 | EP2 | EP1 | EP0 |
| R-0 | | | | | | | Reserved |

**Table 23-9. USB Transmit Interrupt Status Register (USBRXIS) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 15 | EP15 | | RX Endpoint 15 Interrupt |
| | | 0 | No interrupt |
| | | 1 | The Endpoint 15 receive interrupt is asserted. |
| 14 | EP14 | | RX Endpoint 14 Interrupt |
| | | 0 | No interrupt |
| | | 1 | The Endpoint 14 receive interrupt is asserted. |
| 13 | EP13 | | RX Endpoint 13 Interrupt |
| | | 0 | No interrupt |
| | | 1 | The Endpoint 13 receive interrupt is asserted. |
| 12 | EP12 | | RX Endpoint 12 Interrupt |
| | | 0 | No interrupt |
| | | 1 | The Endpoint 12 receive interrupt is asserted. |
| 11 | EP11 | | RX Endpoint 11 Interrupt |
| | | 0 | No interrupt |
| | | 1 | The Endpoint 11 receive interrupt is asserted. |
| 10 | EP10 | | RX Endpoint 10 Interrupt |
| | | 0 | No interrupt |
| | | 1 | The Endpoint 10 receive interrupt is asserted. |
| 9 | EP9 | | RX Endpoint 9 Interrupt |
| | | 0 | No interrupt |
| | | 1 | The Endpoint 9 receive interrupt is asserted. |



Copyright © 2024 Texas Instruments Incorporated

**Table 23-9. USB Transmit Interrupt Status Register (USBRXIS) Field Descriptions (continued)**

| Bit | Field | Value | Description |
|---|---|---|---|
| 8 | EP8 | | RX Endpoint 8 Interrupt |
| | | 0 | No interrupt |
| | | 1 | The Endpoint 8 receive interrupt is asserted. |
| 7 | EP7 | | RX Endpoint 7 Interrupt |
| | | 0 | No interrupt |
| | | 1 | The Endpoint 7 receive interrupt is asserted. |
| 6 | EP6 | | RX Endpoint 6 Interrupt |
| | | 0 | No interrupt |
| | | 1 | The Endpoint 6 receive interrupt is asserted. |
| 5 | EP5 | | RX Endpoint 5 Interrupt |
| | | 0 | No interrupt |
| | | 1 | The Endpoint 5 receive interrupt is asserted. |
| 4 | EP4 | | RX Endpoint 4 Interrupt |
| | | 0 | No interrupt |
| | | 1 | The Endpoint 4 receive interrupt is asserted. |
| 3 | EP3 | | RX Endpoint 3 Interrupt |
| | | 0 | No interrupt |
| | | 1 | The Endpoint 3 receive interrupt is asserted. |
| 2 | EP2 | | RX Endpoint 2 Interrupt |
| | | 0 | No interrupt |
| | | 1 | The Endpoint 2 receive interrupt is asserted. |
| 1 | EP1 | | RX Endpoint 1 Interrupt |
| | | 0 | No interrupt |
| | | 1 | The Endpoint 1 receive interrupt is asserted. |
| 0 | Reserved | | Reserved |



Copyright © 2024 Texas Instruments Incorporated

#### 23.6.3.5 USB Transmit Interrupt Enable Register

#### (USBTXIE), offset 0x006

The USB transmit interrupt enable 16-bit register (USBTXIE) provides interrupt enable bits for the interrupts in the USBTXIS register. When a bit is set, the USB interrupt is asserted to the interrupt controller when the corresponding interrupt bit in the USBTXIS register is set. When a bit is cleared, the interrupt in the USBTXIS register is still set but the USB interrupt to the interrupt controller is not asserted. On reset, all interrupts are enabled.

**Note:** The EP0 bit is special in Host and Device modes. Both the control IN and control OUT endpoints are captured in the EP0 bit of the USBTXIE register.

**Mode(s):** Host Device

USBTXIS is shown in Figure 23-8 and described in Table 23-10.

**Figure 23-8. USB Transmit Interrupt Status Enable Register (USBTXIE)**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| EP15 | EP14 | EP13 | EP12 | EP11 | EP10 | EP9 | EP8 |
| R-0 | | | | | | | |
| EP7 | EP6 | EP5 | EP4 | EP3 | EP2 | EP1 | EP0 |
| R-0 | | | | R/W-1 | R/W-1 | R/W-1 | R/W-1 |

**Table 23-10. USB Transmit Interrupt Status Register (USBTXIE) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 15 | EP15 | | TX Endpoint 15 Interrupt Enable |
| | | 0 | The EP15 transmit interrupt is suppressed and not sent to the interrupt controller. |
| | | 1 | An interrupt is sent to the interrupt controller when the EP15 bit in the USBTXIS register is set. |
| 14 | EP14 | | TX Endpoint 14 Interrupt Enable |
| | | 0 | The EP14 transmit interrupt is suppressed and not sent to the interrupt controller. |
| | | 1 | An interrupt is sent to the interrupt controller when the EP14 bit in the USBTXIS register is set. |
| 13 | EP13 | | TX Endpoint 13 Interrupt Enable |
| | | 0 | The EP13 transmit interrupt is suppressed and not sent to the interrupt controller. |
| | | 1 | An interrupt is sent to the interrupt controller when the EP13 bit in the USBTXIS register is set. |
| 12 | EP12 | | TX Endpoint 12 Interrupt Enable |
| | | 0 | The EP12 transmit interrupt is suppressed and not sent to the interrupt controller. |
| | | 1 | An interrupt is sent to the interrupt controller when the EP12 bit in the USBTXIS register is set. |
| 11 | EP11 | | TX Endpoint 11 Interrupt Enable |
| | | 0 | The EP11 transmit interrupt is suppressed and not sent to the interrupt controller. |
| | | 1 | An interrupt is sent to the interrupt controller when the EP11 bit in the USBTXIS register is set. |
| 10 | EP10 | | TX Endpoint 10 Interrupt Enable |
| | | 0 | The EP10 transmit interrupt is suppressed and not sent to the interrupt controller. |
| | | 1 | An interrupt is sent to the interrupt controller when the EP10 bit in the USBTXIS register is set. |
| 9 | EP9 | | TX Endpoint 9 Interrupt Enable |
| | | 0 | The EP9 transmit interrupt is suppressed and not sent to the interrupt controller. |
| | | 1 | An interrupt is sent to the interrupt controller when the EP9 bit in the USBTXIS register is set. |
| 8 | EP8 | | TX Endpoint 8 Interrupt Enable |
| | | 0 | The EP8 transmit interrupt is suppressed and not sent to the interrupt controller. |
| | | 1 | An interrupt is sent to the interrupt controller when the EP8 bit in the USBTXIS register is set. |



Copyright © 2024 Texas Instruments Incorporated

**Table 23-10. USB Transmit Interrupt Status Register (USBTXIE) Field Descriptions (continued)**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7 | EP7 | | TX Endpoint 7 Interrupt Enable |
| | | 0 | The EP7 transmit interrupt is suppressed and not sent to the interrupt controller. |
| | | 1 | An interrupt is sent to the interrupt controller when the EP7 bit in the USBTXIS register is set. |
| 6 | EP6 | | TX Endpoint 6 Interrupt Enable |
| | | 0 | The EP6 transmit interrupt is suppressed and not sent to the interrupt controller. |
| | | 1 | An interrupt is sent to the interrupt controller when the EP6 bit in the USBTXIS register is set. |
| 5 | EP5 | | TX Endpoint 5 Interrupt Enable |
| | | 0 | The EP5 transmit interrupt is suppressed and not sent to the interrupt controller. |
| | | 1 | An interrupt is sent to the interrupt controller when the EP5 bit in the USBTXIS register is set. |
| 4 | EP4 | | TX Endpoint 4 Interrupt Enable |
| | | 0 | The EP4 transmit interrupt is suppressed and not sent to the interrupt controller. |
| | | 1 | An interrupt is sent to the interrupt controller when the EP4 bit in the USBTXIS register is set. |
| 3 | EP3 | | TX Endpoint 3 Interrupt Enable |
| | | 0 | The EP3 transmit interrupt is suppressed and not sent to the interrupt controller. |
| | | 1 | An interrupt is sent to the interrupt controller when the EP3 bit in the USBTXIS register is set. |
| 2 | EP2 | | TX Endpoint 2 Interrupt Enable |
| | | 0 | The EP2 transmit interrupt is suppressed and not sent to the interrupt controller. |
| | | 1 | An interrupt is sent to the interrupt controller when the EP2 bit in the USBTXIS register is set. |
| 1 | EP1 | | TX Endpoint 1 Interrupt Enable |
| | | 0 | The EP1 transmit interrupt is suppressed and not sent to the interrupt controller. |
| | | 1 | An interrupt is sent to the interrupt controller when the EP1 bit in the USBTXIS register is set. |
| 0 | EP0 | | TX and RX Endpoint 0 Interrupt Enable |
| | | 0 | The EP0 transmit and receive interrupt is suppressed and not sent to the interupt controller. |
| | | 1 | An interrupt is sent to the interrupt controller when the EP0 bit in the USBTXIS register is set. |



Copyright © 2024 Texas Instruments Incorporated

#### 23.6.3.6 USB Receive Interrupt Enable Register

#### (USBRXIE), offset 0x008

The USB receive interrupt enable 16-bit register (USBRXIE) provides interrupt enable bits for the interrupts in the USBRXIS register. When a bit is set, the USB interrupt is asserted to the interrupt controller when the corresponding interrupt bit in the USBRXIS register is set. When a bit is cleared, the interrupt in the USBRXIS register is still set but the USB interrupt to the interrupt controller is not asserted. On reset, all interrupts are enabled.

**Note:** The USBRXIE register does not have a bit for EP0. See the USBTXIE register for EP0 use.

**Mode(s):** Host Device

USBRXIE is shown in Figure 23-9 and described in Table 23-11.

**Figure 23-9. USB Transmit Interrupt Status Enable Register (USBRXIE)**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| EP15 | EP14 | EP13 | EP12 | EP11 | EP10 | EP9 | EP8 |
| R-0 | | | | | | | |
| EP7 | EP6 | EP5 | EP4 | EP3 | EP2 | EP1 | EP0 |
| R-0 | | | | R/W-1 | R/W-1 | R/W-1 | Reserved |

**Table 23-11. USB Transmit Interrupt Status Register (USBRXIE) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 15 | EP15 | | RX Endpoint 15 Interrupt Enable |
| | | 0 | The EP15 transmit interrupt is suppressed and not sent to the interrupt controller. |
| | | 1 | An interrupt is sent to the interrupt controller when the EP15 bit in the USBRXIS register is set. |
| 14 | EP14 | | RX Endpoint 14 Interrupt Enable |
| | | 0 | The EP14 transmit interrupt is suppressed and not sent to the interrupt controller. |
| | | 1 | An interrupt is sent to the interrupt controller when the EP14 bit in the USBRXIS register is set. |
| 13 | EP13 | | RX Endpoint 13 Interrupt Enable |
| | | 0 | The EP13 transmit interrupt is suppressed and not sent to the interrupt controller. |
| | | 1 | An interrupt is sent to the interrupt controller when the EP13 bit in the USBRXIS register is set. |
| 12 | EP12 | | RX Endpoint 12 Interrupt Enable |
| | | 0 | The EP12 transmit interrupt is suppressed and not sent to the interrupt controller. |
| | | 1 | An interrupt is sent to the interrupt controller when the EP12 bit in the USBRXIS register is set. |
| 11 | EP11 | | RX Endpoint 11 Interrupt Enable |
| | | 0 | The EP11 transmit interrupt is suppressed and not sent to the interrupt controller. |
| | | 1 | An interrupt is sent to the interrupt controller when the EP11 bit in the USBRXIS register is set. |
| 10 | EP10 | | RX Endpoint 10 Interrupt Enable |
| | | 0 | The EP10 transmit interrupt is suppressed and not sent to the interrupt controller. |
| | | 1 | An interrupt is sent to the interrupt controller when the EP10 bit in the USBRXIS register is set. |
| 9 | EP9 | | RX Endpoint 9 Interrupt Enable |
| | | 0 | The EP9 transmit interrupt is suppressed and not sent to the interrupt controller. |
| | | 1 | An interrupt is sent to the interrupt controller when the EP9 bit in the USBRXIS register is set. |
| 8 | EP8 | | RX Endpoint 8 Interrupt Enable |
| | | 0 | The EP8 transmit interrupt is suppressed and not sent to the interrupt controller. |
| | | 1 | An interrupt is sent to the interrupt controller when the EP8 bit in the USBRXIS register is set. |



Copyright © 2024 Texas Instruments Incorporated

**Table 23-11. USB Transmit Interrupt Status Register (USBRXIE) Field Descriptions (continued)**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7 | EP7 | | RX Endpoint 7 Interrupt Enable |
| | | 0 | The EP7 transmit interrupt is suppressed and not sent to the interrupt controller. |
| | | 1 | An interrupt is sent to the interrupt controller when the EP7 bit in the USBRXIS register is set. |
| 6 | EP6 | | RX Endpoint 6 Interrupt Enable |
| | | 0 | The EP6 transmit interrupt is suppressed and not sent to the interrupt controller. |
| | | 1 | An interrupt is sent to the interrupt controller when the EP6 bit in the USBRXIS register is set. |
| 5 | EP5 | | RX Endpoint 5 Interrupt Enable |
| | | 0 | The EP5 transmit interrupt is suppressed and not sent to the interrupt controller. |
| | | 1 | An interrupt is sent to the interrupt controller when the EP5 bit in the USBRXIS register is set. |
| 4 | EP4 | | RX Endpoint 4 Interrupt Enable |
| | | 0 | The EP4 transmit interrupt is suppressed and not sent to the interrupt controller. |
| | | 1 | An interrupt is sent to the interrupt controller when the EP4 bit in the USBRXIS register is set. |
| 3 | EP3 | | RX Endpoint 3 Interrupt Enable |
| | | 0 | The EP3 transmit interrupt is suppressed and not sent to the interrupt controller. |
| | | 1 | An interrupt is sent to the interrupt controller when the EP3 bit in the USBRXIS register is set. |
| 2 | EP2 | | RX Endpoint 2 Interrupt Enable |
| | | 0 | The EP2 transmit interrupt is suppressed and not sent to the interrupt controller. |
| | | 1 | An interrupt is sent to the interrupt controller when the EP2 bit in the USBRXIS register is set. |
| 1 | EP1 | | RX Endpoint 1 Interrupt Enable |
| | | 0 | The EP1 transmit interrupt is suppressed and not sent to the interrupt controller. |
| | | 1 | An interrupt is sent to the interrupt controller when the EP1 bit in the USBRXIS register is set. |
| 0 | Reserved | 0 | Reserved |



Copyright © 2024 Texas Instruments Incorporated

#### 23.6.3.7 USB General Interrupt Status Register (USBIS), offset 0x00A

**CAUTION**
Use caution when reading this register. Performing a read may change bit status.

The USB general interrupt status 8-bit read-only register (USBIS) indicates which USB interrupts are currently active. All active interrupts are cleared when this register is read.

**Mode(s):** Host Device

USBIS in Host Mode is shown in Figure 23-10 and described in Table 23-12.

**Figure 23-10. USB General Interrupt Status Register (USBIS) in Host Mode**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| VBUSERR | SESREQ | DISCON | CONN | SOF | BABBLE | RESUME | Reserved |
| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |

LEGEND: R = Read only; *-n* = value after reset

**Table 23-12. USB General Interrupt Status Register (USBIS) in Host Mode Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7 | VBUSERR | | VBUS Error |
| | | 0 | No interrupt |
| | | 1 | VBUS has dropped below the VBUS Valid threshold during a session. |
| 6 | SESREQ | | Session Request |
| | | 0 | No interrupt |
| | | 1 | SESSION REQUEST signaling has been detected. |
| 5 | DISCON | | Session Disconnect |
| | | 0 | No interrupt |
| | | 1 | A Device disconnect has been detected. |
| 4 | CONN | | Session Connect |
| | | 0 | No interrupt |
| | | 1 | A Device connection has been detected. |
| 3 | SOF | | Start of Frame |
| | | 0 | No interrupt |
| | | 1 | A new frame has started. |
| 2 | BABBLE | | Babble Detected |
| | | 0 | No interrupt |
| | | 1 | Babble has been detected. This interrupt is active only after the first SOF has been sent. |
| 1 | RESUME | | RESUME Signaling Detected. This interrupt can only be used if the USB controller's system clock is enabled. If the user disables the clock programming, the USBDRRIS, USBDRIM, and USBDRISC registers should be used. |
| | | 0 | No effect |
| | | 1 | RESUME signaling has been detected on the bus while the USB controller is in SUSPEND mode. |
| 0 | Reserved | 0 | Reserved |



Copyright © 2024 Texas Instruments Incorporated

USBIS in Device Mode is shown in Figure 23-11 and described in Table 23-13.

**Figure 23-11. USB General Interrupt Status Register (USBIS) in Device Mode**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | DISCON | Reserved | SOF | RESET | RESUME | SUSPEND |
| R-0 | | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |

LEGEND: R = Read only; *-n* = value after reset

**Table 23-13. USB General Interrupt Status Register (USBIS) in Device Mode Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7-6 | Reserved | 0 | Reserved |
| 5 | DISCON | | Session Disconnect |
| | | 0 | No interrupt |
| | | 1 | The device has been disconnected from the host. |
| 4 | Reserved | 0 | Reserved |
| 3 | SOF | | Start of frame |
| | | 0 | No interrupt |
| | | 1 | A new frame has started. |
| 2 | RESET | | RESET Signaling Detected |
| | | 0 | No interrupt |
| | | 1 | RESET signaling has been detected on the bus. |
| 1 | RESUME | | RESUME Signaling Detected. This interrupt can only be used if the USB controller's system clock is enabled. If the user disables the clock programming, the USBDRRIS, USBDRIM, and USBDRISC registers should be used. |
| | | 0 | No interrupt |
| | | 1 | RESUME signaling has been detected on the bus while the USB controller is in SUSPEND mode. |
| 0 | SUSPEND | | SUSPEND Signaling Detected |
| | | 0 | No interrupt |
| | | 1 | SUSPEND signaling has been detected on the bus. |



Copyright © 2024 Texas Instruments Incorporated

### 23.6.3.8 USB Interrupt Enable Register (USBIE), offset 0x00B

**CAUTION**
Use caution when reading this register. Performing a read may change bit status.

The USB interrupt enable 8-bit register (USBIE) provides interrupt enable bits for each of the interrupts in USBIS. At reset interrupts 1 and 2 are enabled in device mode.

**Mode(s):** Host Device

USBIE in Host Mode is shown in Figure 23-12 and described in Table 23-14.

**Figure 23-12. USB Interrupt Enable Register (USBIE) in Host Mode**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| VBUSERR | SESREQ | DISCON | CONN | SOF | BABBLE | RESUME | Reserved |
| R-W | R-W | R-W | R-W | R-W | R-W | R-W | R-0 |

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

**Table 23-14. USB Interrupt Enable Register (USBIE) in Host Mode Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7 | VBUSERR | | Enable VBUS Error Interrupt |
| | | 0 | The VBUSERR interrupt is suppressed and not sent to the interrupt controller. |
| | | 1 | An interrupt is sent to the interrupt controller when the VBUSERR bit in the USBIS register is set. |
| 6 | SESREQ | | Enable Session Request |
| | | 0 | The SESREQ interrupt is suppressed and not sent to the interrupt controller. |
| | | 1 | An interrupt is sent to the interrupt controller when the SESREEQ bit in the USBIS register is set. |
| 5 | DISCON | | Enable Disconnect Interrupt |
| | | 0 | The DISCON interrupt is suppressed and not sent to the interrupt controller. |
| | | 1 | An interrupt is sent to the interrupt controller when the DISCON bit in the USBIS register is set. |
| 4 | CONN | | Enable Connect Interrupt |
| | | 0 | The CONN interrupt is suppressed and not sent to the interrupt controller. |
| | | 1 | An interrupt is sent to the interrupt controller when the CONN bit in the USBIS register is set. |
| 3 | SOF | | Start of Frame |
| | | 0 | The SOF interrupt is suppressed and not sent to the interrupt controller. |
| | | 1 | An interrupt is sent to the interrupt controller when the SOF bit in the USBIS register is set. |
| 2 | BABBLE | | Babble Detected |
| | | 0 | The BABBLE interrupt is suppressed and not sent to the interrupt controller. |
| | | 1 | An interrupt is sent to the interrupt controller when the BABBLE bit in the USBIS register is set. |
| 1 | RESUME | | RESUME Signaling Detected. This interrupt can only be used if the USB controller's system clock is enabled. If the user disables the clock programming, the USBDRRIS, USBDRIM, and USBDRISC registers should be used. |
| | | 0 | The RESUME interrupt is suppressed and not sent to the interrupt controller. |
| | | 1 | An interrupt is sent to the interrupt controller when the RESUME bit in the USBIS register is set. |
| 0 | Reserved | 0 | Reserved |



Copyright © 2024 Texas Instruments Incorporated

USBIE in Device Mode is shown in Figure 23-11 and described in Table 23-13.

**Figure 23-13. USB Interrupt Enable Register (USBIE) in Device Mode**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | DISCON | Reserved | SOF | RESET | RESUME | SUSPEND |
| R-0 | | R/W-0 | R-0 | R/W-0 | R/W-1 | RW-1 | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

**Table 23-15. USB Interrupt Enable Register (USBIE) in Device Mode Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7-6 | Reserved | 0 | Reserved |
| 5 | DISCON | | Enable Disconnect Interrupt |
| | | 0 | The DISCON interrupt is suppressed and not sent to the interrupt controller. |
| | | 1 | An interrupt is sent to the interrupt controller when the DISCON bit in the USBIS register is set. |
| 4 | Reserved | 0 | Reserved |
| 3 | SOF | | Start of frame |
| | | 0 | The SOF interrupt is suppressed and not sent to the interrupt controller. |
| | | 1 | An interrupt is sent to the interrupt controller when the SOF bit in the USBIS register is set. |
| 2 | RESET | | RESET Signaling Detected |
| | | 0 | The RESET interrupt is suppressed and not sent to the interrupt controller. |
| | | 1 | An interrupt is sent to the interrupt controller when the RESET bit in the USBIS register is set. |
| 1 | RESUME | | RESUME Signaling Detected. This interrupt can only be used if the USB controller's system clock is enabled. If the user disables the clock programming, the USBDRRIS, USBDRIM, and USBDRISC registers should be used. |
| | | 0 | The RESUME interrupt is suppressed and not sent to the interrupt controller. |
| | | 1 | An interrupt is sent to the interrupt controller when the RESUME bit in the USBIS register is set. |
| 0 | SUSPEND | | SUSPEND Signaling Detected |
| | | 0 | The SUSPEND interrupt is suppressed and not sent to the interrupt controller. |
| | | 1 | An interrupt is sent to the interrupt controller when the DISCON bit in the USBIS register is set. |



Copyright © 2024 Texas Instruments Incorporated

#### 23.6.3.9 USB Frame Value Register (USBFRAME), offset 0x00C

The frame number 16-bit read-only register (USBFRAME) holds the last received frame number.

**Mode(s):** Host Device

USBFRAME is shown in Figure 23-14 and described in Table 23-16.

**Figure 23-14. USB Frame Value Register (USBFRAME)**

| 15 | 11 | 10 | 0 |
|---|---|---|---|
| Reserved | | FRAME | |
| R-0 | | R-0 | |

LEGEND: R = Read only; *-n* = value after reset

**Table 23-16. USB Frame Value Register (USBFRAME) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 15-11 | Reserved | 0 | Reserved |
| 10-0 | FRAME | 0-7FFh | Last received frame number |

#### 23.6.3.10 USB Endpoint Index Register (USBEPIDX), offset 0x00E

Each endpoint buffer can be accessed by configuring a FIFO size and starting address. The endpoint index 16-bit register (USBEPIDX) is used with the USBTXFIFOSZ, USBRXFIFOSZ, USBTXFIFOADD, and USBRXFIFOADD registers.

**Mode(s):** Host Device

USBEPIDX is shown in Figure 23-15 and described in Table 23-17.

**Figure 23-15. USB Endpoint Index Register (USBEPIDX)**

| 7 | 4 | 3 | 0 |
|---|---|---|---|
| Reserved | | EPIDX | |
| R-0 | | R/W-0 | |

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

**Table 23-17. USB Endpoint Index Register (USBEPIDX) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7-4 | Reserved | 0 | Reserved |
| 3-0 | EPIDX | 0-4h | Endpoint Index. This bit field configures which endpoint is accessed when reading or writing to one of the USB controller's indexed registers. A value of 0x0 corresponds to Endpoint 0 and a value of 0xF corresponds to Endpoint 15. |



Copyright © 2024 Texas Instruments Incorporated

#### 23.6.3.11 USB Test Mode Register (USBTEST), offset 0x00F

The USB test mode 8-bit register (USBTEST) is primarily used to put the USB controller into one of the four test modes for operation described in the USB Specification 2.0 , in response to a SET FEATURE: USBTESTMODE command. This register is not used in normal operation.

**Note:** Only one of these bits should be set at any time.

**Mode(s):** Host Device

USBTEST in Host Mode is shown in Figure 23-16 and described in Table 23-18.

**Figure 23-16. USB Test Mode Register (USBTEST) in Host Mode**

| 7 | 6 | 5 | 4 | 0 |
|---|---|---|---|---|
| FORCEH | FIFOACC | FORCEFS | Reserved | |
| R/W-0 | R/W1S-0 | R/W-0 | R-0 | |

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

**Table 23-18. USB Test Mode Register (USBTEST) in Host Mode Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7 | FORCEH | | Force Host Mode. While in this mode, status of the bus connection may be read using the DEV bit of the USBDEVCTL register. The operating speed is determined from the FORCEFS bit. |
| | | 0 | No effect |
| | | 1 | Forces the USB controller to enter Host mode when the SESSION bit is set, regardless of whether the USB controller is connected to any peripheral. The state of the USB0DP and USB0DM signals is ignored. The USB controller then remains in Host mode until the SESSION bit is cleared, even if a Device is disconnected. If the FORCEH bit remains set, the USB controller re-enters Host mode the next time the SESSION bit is set. |
| 6 | FIFOACC | | FIFO Access |
| | | 0 | No effect |
| | | 1 | Transfers the packet in the endpoint 0 transmit FIFO to the endpoint 0 receive FIFO. |
| 5 | FORCEFS | | Force Full-Speed Mode |
| | | 0 | The USB controller operates at Low Speed. |
| | | 1 | Forces the USB controller into Full-Speed mode upon receiving a USB RESET. |
| 4-0 | Reserved | 0 | Reserved |



Copyright © 2024 Texas Instruments Incorporated

USBTEST in Device Mode is shown in Figure 23-17 and described in Table 23-19.

**Figure 23-17. USB Test Mode Register (USBTEST) in Device Mode**

| 7 | 6 | 5 | 4 | 0 |
|---|---|---|---|---|
| Reserved | FIFOACC | FORCEFS | Reserved | |
| R-0 | R/W1S-0 | R/W-0 | R-0 | |

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

**Table 23-19. USB Test Mode Register (USBTEST) in Device Mode Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7 | Reserved | | Force Host Mode. While in this mode, status of the bus connection may be read using the DEV bit of the USBDEVCTL register. The operating speed is determined from the FORCEFS bit. |
| 6 | FIFOACC | | FIFO Access |
| | | 0 | No effect |
| | | 1 | Transfers the packet in the endpoint 0 transmit FIFO to the endpoint 0 receive FIFO. |
| 5 | FORCEFS | | Force Full-Speed Mode |
| | | 0 | The USB controller operates at Low Speed. |
| | | 1 | Forces the USB controller into Full-Speed mode upon receiving a USB RESET. |
| 4-0 | Reserved | 0 | Reserved |



Copyright © 2024 Texas Instruments Incorporated

#### 23.6.3.12 USB FIFO Endpoint n Register (USBFIFO[0]-USBFIFO[3])

**Note**

Use caution when reading these registers. Performing a read may change bit status.

The USB FIFO endpoint *n* 32-bit registers (USBFIFO[*n*]) provide an address for CPU access to the FIFOs for each endpoint. Writing to these addresses loads data into the Transmit FIFO for the corresponding endpoint. Reading from these addresses unloads data from the Receive FIFO for the corresponding endpoint.

Transfers to and from FIFOs can be 8-bit, 16-bit or 32-bit as required, and any combination of accesses is allowed provided the data accessed is contiguous. All transfers associated with one packet must be of the same width so that the data is consistently byte-, halfword- or word-aligned. However, the last transfer may contain fewer bytes than the previous transfers in order to complete an odd-byte or odd-word transfer.

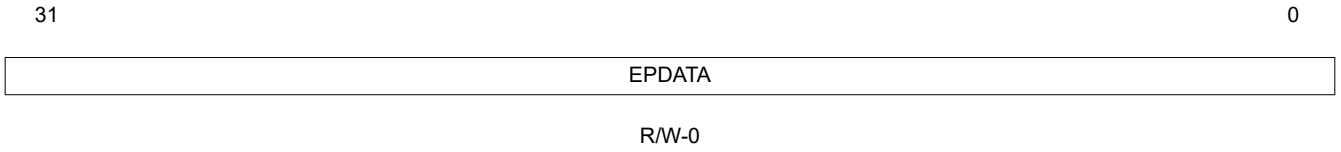
Depending on the size of the FIFO and the expected maximum packet size, the FIFOs support either single-packet or double-packet buffering (see *Single-Packet Buffering* in Section 23.2.1.1.2). Burst writing of multiple packets is not supported as flags must be set after each packet is written.

Following a STALL response or a transmit error on endpoint 1–3, the associated FIFO is completely flushed.

**Mode(s):** Host Device

USBFIFO0-3 are shown in Figure 23-18 and described in Table 23-20.

**Figure 23-18. USB FIFO Endpoint *n* Register (USBFIFO[*n*])**




LEGEND: R/W = Read/Write; -*n* = value after reset

**Table 23-20. USB FIFO Endpoint *n* Register (USBFIFO[*n*]) Field Descriptions**

| Bit | Field | Reset | Description |
|---|---|---|---|
| 31-0 | EPDATA | 0x0000.0000 | Endpoint Data. Writing to this register loads the data into the Transmit FIFO and reading unloads data from the Receive FIFO. |



Copyright © 2024 Texas Instruments Incorporated

#### 23.6.3.13 USB Device Control Register (USBDEVCTL), offset 0x060

The USB device control 8-bit register (USBDEVCTL) is used for controlling and monitoring the USB VBUS line. If the PHY is suspended, no PHY clock is received and the VBUS is not sampled. In addition, in Host mode, USBDEVCTL provides the status information for the current operating mode (Host or Device) of the USB controller. If the USB controller is in Host mode, this register also indicates if a full- or low-speed Device has been connected.

**Mode(s):** Host Device

USBDEVCTL is shown in Figure 23-19 and described in Table 23-21.

**Figure 23-19. USB Device Control Register (USBDEVCTL)**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| DEV | FSDEV | LSDEV | VBUS | | HOSTMODE | HOSTREQ | SESSION |
| R-1 | R-0 | R-0 | R-0 | | R-0 | R/W-0 | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

**Table 23-21. USB Device Control Register (USBDEVCTL) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7 | DEV | | Device mode |
| | | 0 | The USB controller is operating on the OTG A side of the cable. |
| | | 1 | The USB controller is operating on the OTG B side of the cable. |
| | | | Only valid while a session is in progress. |
| 6 | FSDEV | | Full-Speed Device Detected |
| | | 0 | A full-speed Device has not been detected on the port. |
| | | 1 | A full-speed Device has been detected on the port. |
| 5 | LSDEV | | Low-Speed Device Detected |
| | | 0 | A low-speed Device has not been detected on the port. |
| | | 1 | A low-speed Device has been detected on the port. |
| 4-3 | VBUS | 0-3h | These read-only bits encode the current VBus level as follows: |
| | | 0 | Below Session End. VBUS is detected as under 0.5 V. |
| | | 1h | Above Session End, below AValid. VBUS is detected as above 0.5 V and under 1.5 V. |
| | | 2h | Above AValid, below VBusValid. VBUS is detected as above 1.5 V and below 4.75 V. |
| | | 3h | Above VBusValid. VBUS is detected as above 4.75 V. |
| 2 | HOSTMODE | | This read-only bit is set when the USB controller is acting as a Host. |
| | | 0 | The USB controller is acting as a Device. |
| | | 1 | The USB controller is acting as a Host. |
| | | | Only valid while a session is in progress. |
| 1 | HOSTREQ | | When set, the USB controller will initiate the Host Negotiation when Suspend mode is entered. It is cleared when Host Negotiation is completed. |
| | | 0 | No effect |
| | | 1 | Initiates the Host Negotiation when SUSPENDmode is entered. |



Copyright © 2024 Texas Instruments Incorporated

**Table 23-21. USB Device Control Register (USBDEVCTL) Field Descriptions (continued)**

| Bit | Field | Value | Description |
|---|---|---|---|
| 0 | SESSION | | Session Start/End<br>When operating as a Host: |
| | | 0 | When cleared by software, this bit ends a session. |
| | | 1 | When set by software, this bit starts a session. |
| | | | When operating as a Device: |
| | | 0 | The USB controller has ended a session. When the USB controller is in SUSPEND mode, this bit may be cleared by software to perform a software disconnect. |
| | | 1 | The USB controller has started a session. When set by software, the Session Request Protocol is initiated. |
| | | | Clearing this bit when the USB controller is not suspended results in undefined behavior. |



Copyright © 2024 Texas Instruments Incorporated

#### 23.6.3.14 USB Transmit Dynamic FIFO Sizing Register (USBTXFIFOSZ), offset 0x062

The USB transmit dynamic FIFO sizing 8-bit register (USBTXFIFOSZ) allows the selected TX endpoint FIFOs to be dynamically sized. USBEPIDX is used to configure each transmit endpoint's FIFO size.

**Mode(s):** Host Device

USBTXFIFOSZ is shown in Figure 23-20 and described in Table 23-22.

**Figure 23-20. USB Transmit Dynamic FIFO Sizing Register (USBTXFIFOSZ)**

| 7 | 5 | 4 | 3 | 0 |
|---|---|---|---|---|
| Reserved | | DPB | SZ | |
| R-0 | | R/W-0 | R-0 | |

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

**Table 23-22. USB Transmit Dynamic FIFO Sizing Register (USBTXFIFOSZ) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7-5 | Reserved | 0 | Reserved |
| 4 | DPB | | Double Packet Buffering Support |
| | | 0 | Single packet buffering is supported. |
| | | 1 | Double packet buffering is enabled. |
| 3-0 | SZ | | Maximum packet size to be allowed. If DPB = 0, the FIFO also is this size; if DPB = 1, the FIFO is twice this size. Packet size in bytes: |
| | | 0h | 8 |
| | | 1h | 16 |
| | | 2h | 32 |
| | | 3h | 64 |
| | | 4h | 128 |
| | | 5h | 256 |
| | | 6h | 512 |
| | | 7h | 1024 |
| | | 8h | 2048 |
| | | 9-Fh | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 23.6.3.15 USB Receive Dynamic FIFO Sizing Register (USBRXFIFOSZ), offset 0x063

The USB receive dynamic FIFO sizing 8-bit register (USBRXFIFOSZ) allows the selected RX endpoint FIFOs to be dynamically sized.

**Mode(s):** Host Device

USBRXFIFOSZ is shown in Figure 23-21 and described in Table 23-23.

**Figure 23-21. USB Receive Dynamic FIFO Sizing Register (USBRXFIFOSZ)**

| 7 | 5 | 4 | 3 | 0 |
|---|---|---|---|---|
| Reserved | | DPB | SZ | |
| R-0 | | R/W-0 | R-0 | |

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

**Table 23-23. USB Receive Dynamic FIFO Sizing Register (USBRXFIFOSZ) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7-5 | Reserved | 0 | Reserved |
| 4 | DPB | | Double Packet Buffering Support |
| | | 0 | Single packet buffering is supported. |
| | | 1 | Double packet buffering is enabled. |
| 3-0 | SZ | | Maximum packet size to be allowed. If DPB = 0, the FIFO also is this size; if DPB = 1, the FIFO is twice this size. Packet size in bytes: |
| | | 0h | 8 |
| | | 1h | 16 |
| | | 2h | 32 |
| | | 3h | 64 |
| | | 4h | 128 |
| | | 5h | 256 |
| | | 6h | 512 |
| | | 7h | 1024 |
| | | 8h | 2048 |
| | | 9-Fh | Reserved |



Copyright © 2024 Texas Instruments Incorporated

#### 23.6.3.16 USB Transmit FIFO Start Address Register (USBTXFIFOADD), offset 0x064

The USB transmit FIFO start address 16-bit register (USBTXFIFOADD) controls the start address of the selected transmit endpoint FIFOs.

**Mode(s):** Host Device

USBTXFIFOADDR is shown in Figure 23-22 and described in Table 23-24.

**Figure 23-22. USB Transmit FIFO Start Address Register (USBTXFIFOADDR])**

| 15 | 9 | 8 | 0 |
|---|---|---|---|
| Reserved | | ADDR | |
| R-0 | | R/W-0 | |

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

**Table 23-24. USB Transmit FIFO Start Address Register (USBTXFIFOADDR) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 15-9 | Reserved | 0 | Reserved |
| 8-0 | ADDR | | Start Address of the endpoint FIFO in units of 8 bytes. |
| | | 0h | 0 |
| | | 1h | 8 |
| | | 2h | 16 |
| | | 3h | 24 |
| | | 4h | 32 |
| | | 5h | 40 |
| | | 6h | 48 |
| | | 7h | 56 |
| | | 8h | 64 |
| | | .. | .. |
| | | 1FFh | 4095 |



Copyright © 2024 Texas Instruments Incorporated

### 23.6.3.17 USB Receive FIFO Start Address Register (USBRXFIFOADD), offset 0x066

The USB receive FIFO start address 16-bit register (USBRXFIFOADD) controls the start address of the selected receive endpoint FIFOs.

**Mode(s):** Host Device

USBRXFIFOADDR is shown in Figure 23-23 and described in Table 23-25.

**Figure 23-23. USB Receive FIFO Start Address Register (USBRXFIFOADDR)**

| 15 | 9 | 8 | 0 |
|---|---|---|---|
| Reserved | | ADDR | |
| R-0 | | R/W-0 | |

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

**Table 23-25. USB Receive FIFO Start Address Register (USBRXFIFOADDR) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 15-9 | Reserved | 0 | Reserved |
| 8-0 | ADDR | | Start Address of the endpoint FIFO in units of 8 bytes. |
| | | 0h | 0 |
| | | 1h | 8 |
| | | 2h | 16 |
| | | 3h | 24 |
| | | 4h | 32 |
| | | 5h | 40 |
| | | 6h | 48 |
| | | 7h | 56 |
| | | 8h | 64 |
| | | .. | .. |
| | | 1FFh | 4095 |



Copyright © 2024 Texas Instruments Incorporated

#### 23.6.3.18 USB Connect Timing Register (USBCONTIM), offset 0x07A

The USB connect timing 8-bit configuration register (USBCONTIM) specifies connection and negotiation delays.

**Mode(s):** Host Device

USBCONTIM is shown in Figure 23-24 and described in Table 23-26.

**Figure 23-24. USB Connect Timing Register (USBCONTIM)**

| 7 | 4 | 3 | 0 |
|---|---|---|---|
| WTCON | | WTID | |
| R/W-1 | | R/W-1 | |

LEGEND: R/W = Read/Write; -*n* = value after reset

**Table 23-26. USB Connect Timing Register (USBCONTIM) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7-4 | WTCON | 5h | The connect wait field configures the wait required to allow for the user's connect/disconnect filter, in units of 533.3 ns. The default corresponds to 2.667 µs. |
| 3-0 | WTID | Ch | The wait ID field configures the delay required from the enable of the ID detection to when the ID value is valid, in units of 4.369 ms. The default corresponds to 52.43 ms. |



Copyright © 2024 Texas Instruments Incorporated

#### 23.6.3.19 USB Full-Speed Last Transaction to End of Frame Timing Register (USBFSEOF), offset 0x07D

USB full-speed last transaction to end of frame timing 8-bit configuration register (USBFSEOF) specifies the minimum time gap allowed between the start of the last transaction and the EOF for full-speed transactions.

**Mode(s):** Host Device

USBFSEOF is shown in Figure 23-25 and described in Table 23-27.

**Figure 23-25. USB Full-Speed Last Transaction to End of Frame Timing Register (USBFSEOF)**

| 7 | 0 |
|---|---|
| FSEOFG | |
| R/W-0x77 | |

LEGEND: R/W = Read/Write; *-n* = value after reset

**Table 23-27. USB Full-Speed Last Transaction to End of Frame Timing Register (USBFSEOF) Field Descriptions**

| Bit | Field | Reset | Description |
|---|---|---|---|
| 7-0 | FSEOFG | 77h | The full-speed end-of-frame gap field is used during full-speed transactions to configure the gap between the last transaction and the End-of-Frame (EOF), in units of 533.3 ns. The default corresponds to 63.46 µs. |

#### 23.6.3.20 USB Low-Speed Last Transaction to End of Frame Timing Register (USBLSEOF), offset 0x07E

The USB low-speed last transaction to end of frame timing 8-bit configuration register (USBLSEOF) specifies the minimum time gap that is to be allowed between the start of the last transaction and the EOF for low-speed transactions.

**Mode(s):** Host Device

USBLSEOF is shown in Figure 23-26 and described in Table 23-28.

**Figure 23-26. USB Low-Speed Last Transaction to End of Frame Timing Register (USBLSEOF)**

| 7 | 0 |
|---|---|
| LSEOFG | |
| R/W-0x72 | |

LEGEND: R/W = Read/Write; *-n* = value after reset

**Table 23-28. USB Low-Speed Last Transaction to End of Frame Timing Register (USBLSEOF) Field Descriptions**

| Bit | Field | Reset | Description |
|---|---|---|---|
| 7-0 | LSEOFG | 72h | The low-speed end-of-frame gap field is used during low-speed transactions to set the gap between the last transaction and the End-of-Frame (EOF), in units of 1.067 µs. The default corresponds to 121.6 µs. |



Copyright © 2024 Texas Instruments Incorporated

#### 23.6.3.21 USB Transmit Functional Address Endpoint n Registers (USBTXFUNCADDR[0]-USBTXFUNCADDR[*n*])

The transmit functional address endpoint *n* 8-bit registers (USBTXFUNCADDR[*n*]) record the address of the target function to be accessed through the associated endpoint (EP*n*). USBTXFUNCADDRx must be defined for each transmit endpoint that is used.

**Note:** USBTXFUNCADDR0 is used for both receive and transmit for endpoint 0.

**Mode(s):** Host

The USBTXFUNCADDR[*n*] registers are shown in Figure 23-27 and described in Table 23-29.

**Figure 23-27. USB Transmit Functional Address Endpoint *n* Registers (USBTXFUNCADDR[*n*])**

| 7 | 6 | 0 |
|---|---|---|
| Reserved | ADDR | |
| R-0 | R/W-0 | |

LEGEND: R/W = Read/Write; -*n* = value after reset

**Table 23-29. USB Transmit Functional Address Endpoint *n* Registers (USBTXFUNCADDR[*n*]) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7 | Reserved | 0 | Reserved |
| 6-0 | ADDR | 0 | Device Address specifies the USB bus address for the target Device. |



Copyright © 2024 Texas Instruments Incorporated

#### 23.6.3.22 USB Transmit Hub Address Endpoint n Registers (USBTXHUBADDR[0]-USBTXHUBADDR[*n*])

The transmit hub address endpoint *n* 8-bit read/write registers (USBTXHUBADDR[*n*]), like USBTXHUBPORT[*n*], must be written only when a USB device is connected to transmit endpoint EP*n* via a USB 2.0 hub. This register records the address of the USB 2.0 hub through which the target associated with the endpoint is accessed.

**Note:** USBTXHUBADDR0 is used for both receive and transmit for endpoint 0.

**Mode(s):** Host

The USBTXHUBADDR[*n*] registers are shown in Figure 23-27 and described in Table 23-29.

**Figure 23-28. USB Transmit Hub Address Endpoint *n* Registers (USBTXHUBADDR[*n*])**

| 7 | 6 | 0 |
|---|---|---|
| Reserved | ADDR | |
| R-0 | R/W-0 | |

LEGEND: R/W = Read/Write; -*n* = value after reset

**Table 23-30. USB Transmit Hub Address Endpoint *n* Registers(USBTXHUBADDR[*n*]) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7 | Reserved | 0 | Reserved |
| 6-0 | ADDR | 0 | Device Address specifies the USB bus address for the target Device. |



Copyright © 2024 Texas Instruments Incorporated

#### 23.6.3.23 USB Transmit Hub Port Endpoint n Registers (USBTXHUBPORT[0]-USBTXHUBPORT[*n*])

The transmit hub port endpoint *n* 8-bit read/write registers (USBTXHUBPORT[*n*]), like USBTXHUBADDR[*n*], must be written only when a full- or low-speed Device is connected to transmit endpoint EP*n* with a USB 2.0 hub. This register records the port of the USB 2.0 hub through which the target associated with the endpoint is accessed.

**Note:** USBTXHUBPORT0 is used for both receive and transmit for endpoint 0.

**Mode(s):** Host

The USBTXHUBPORT*n* registers are shown in Figure 23-29 and described in Table 23-31.

**Figure 23-29. USB Transmit Hub Port Endpoint *n* Registers (USBTXHUBPORT[*n*])**

| 7 | 6 | 0 |
|---|---|---|
| Reserved | PORT | |
| R-0 | R/W-0 | |

LEGEND: R/W = Read/Write; -*n* = value after reset

**Table 23-31. USB Transmit Hub Port Endpoint *n* Registers(USBTXHUBPORT[*n*]) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7 | Reserved | 0 | Reserved |
| 6-0 | PORT | 0 | Hub Port specifies the USB hub port number. |



Copyright © 2024 Texas Instruments Incorporated

#### 23.6.3.24 USB Receive Functional Address Endpoint n Registers (USBRXFUNCADDR[1]-USBRXFUNCADDR[*n*])

The receive functional address endpoint *n* 8-bit read/write registers (USBRXFUNCADDR[*n*]) record the address of the target function to be accessed through the associated endpoint (EP*n*). USBRXFUNCADDRx must be defined for each receive endpoint that is used.

**Note:** USBTXFUNCADDR0 is used for both receive and transmit for endpoint 0.

**Mode(s):** Host

The USBRXFUNCADDR[*n*] registers are shown in Figure 23-30 and described in Table 23-32.

**Figure 23-30. USB Receive Functional Address Endpoint *n* Registers (USBFIFO[*n*])**

| 7 | 6 | 0 |
|---|---|---|
| Reserved | ADDR | |
| R-0 | R/W-0 | |

LEGEND: R/W = Read/Write; -*n* = value after reset

**Table 23-32. USB Receive Functional Address Endpoint *n* Registers(USBFIFO[*n*]) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7 | Reserved | 0 | Reserved |
| 6-0 | ADDR | 0 | Device Address specifies the USB bus address for the target Device. |



Copyright © 2024 Texas Instruments Incorporated

#### 23.6.3.25 USB Receive Hub Address Endpoint n Registers (USBRXHUBADDR[1]-USBRXHUBADDR[*n*])

The receive hub address endpoint *n* 8-bit read/write registers (USBRXHUBADDR[*n*]), like [*n*], must be written only when a full- or low-speed Device is connected to receive endpoint EP*n* with a USB 2.0 hub. Each register records the address of the USB 2.0 hub through which the target associated with the endpoint is accessed.

**Note:** USBTXHUBADDR0 is used for both receive and transmit for endpoint 0.

**Mode(s):** Host

The USBRXHUBADDR[*n*] registers are shown in Figure 23-31 and described in Table 23-33.

**Figure 23-31. USB Receive Hub Address Endpoint *n* Registers (USBRXHUBADDR[*n*])**

| 7 | 6 | 0 |
|---|---|---|
| MULTTRAN | ADDR | |
| R/w-0 | R/W-0 | |

LEGEND: R/W = Read/Write; -*n* = value after reset

**Table 23-33. USB Receive Hub Address Endpoint *n* Registers (USBRXHUBADDR[*n*]) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7 | MULTTRAN | | Multiple Translators |
| | | 0 | Clear to indicate that the hub has a single transaction translator. |
| | | 1 | Set to indicate that the hub has multiple transaction translators. |
| 6-0 | ADDR | 0 | Device Address specifies the USB bus address for the target Device. |



Copyright © 2024 Texas Instruments Incorporated

#### 23.6.3.26 USB Receive Hub Port Endpoint n Register (USBRXHUBPORT[1]-USBRXHUBPORT[*n*])

The receive hub port endpoint *n* 8-bit read/write registers (USBRXHUBPORT[*n*]), like USBRXHUBADDR[*n*], must be written only when a full- or low-speed device is connected to receive endpoint EP*n* with a USB 2.0 hub. Each register records the port of the USB 2.0 hub through which the target associated with the endpoint is accessed.

**Note:** USBTXHUBPORT0 is used for both receive and transmit for endpoint 0.

**Mode(s):** Host

The USBRXHUBPORT*n* registers are shown in Figure 23-32 and described in Table 23-34.

**Figure 23-32. USB Receive Hub Port Endpoint *n* Register (USBRXHUBPORT[*n*])**

| 7 | 6 | 0 |
|---|---|---|
| Reserved | PORT | |
| R-0 | R/W-0 | |

LEGEND: R/W = Read/Write; -*n* = value after reset

**Table 23-34. USB Receive Hub Port Endpoint *n* Register (USBRXHUBPORT[*n*]) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7 | Reserved | 0 | Reserved |
| 6-0 | PORT | 0 | Hub Port specifies the USB hub port number. |



Copyright © 2024 Texas Instruments Incorporated

### 23.6.3.27 USB Control and Status Endpoint 0 Low Register (USBCSRL0), offset 0x102

The USB control and status endpoint 0 low 8-bit register (USBCSRL0) provides control and status bits for endpoint 0.

**Mode(s):** Host Device

USBCSRL0 in Host mode is shown in Figure 23-33 and described in Table 23-35.

**Figure 23-33. USB Control and Status Endpoint 0 Low Register (USBCSRL0) in Host Mode**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| NAKTO | STATUS | REQPKT | ERROR | SETUP | STALLED | TXRDY | RXRDY |
| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |

LEGEND: R/W = Read/Write; *-n* = value after reset

**Table 23-35. USB Control and Status Endpoint 0 Low Register (USBCSRL0) in Host Mode Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7 | NAKTO | | NAK Timeout. Software must clear this bit to allow the endpoint to continue. |
| | | 0 | No timeout |
| | | 1 | Indicates that endpoint 0 is halted following the receipt of NAK responses for longer than the time set by the USBNAKLMT register. |
| 6 | STATUS | | Status Packet. Setting this bit ensures that the DT bit is set in the USBCSRH0 register so that a DATA1 packet is used for the STATUS stage transaction. |
| | | 0 | No transaction |
| | | 1 | Initiates a STATUS stage transaction. This bit must be set at the same time as the TXRDY or REQPKT bit is set. |
| | | | This bit is automatically cleared when the STATUS stage is over. |
| 5 | REQPKT | | Request Packet. This bit is cleared when the RXRDY bit is set. |
| | | 0 | No request |
| | | 1 | Requests an IN transaction. |
| 4 | ERROR | | Error. Software must clear this bit. |
| | | 0 | No error |
| | | 1 | Three attempts have been made to perform a transaction with no response from the peripheral. The EP0 bit in the USBTXIS register is also set in this situation. |
| 3 | SETUP | | Setup Packet. Setting this bit always clears the DT bit in the USBCSRH0 register to send a DATA0 packet. |
| | | 0 | Sends an OUT token. |
| | | 1 | Sends a SETUP token instead of an OUT token for the transaction. This bit should be set at the same time as the TXRDY bit is set. |
| 2 | STALLED | | Endpoint Stalled. Software must clear this bit. |
| | | 0 | No handshake has been received. |
| | | 1 | A STALL handshake has been received. |
| 1 | TXRDY | | Transmit Packet Ready. If both the TXRDY and SETUP bits are set, a setup packet is sent. If just TXRDY is set, an OUT packet is sent. |
| | | 0 | No transmit packet is ready. |
| | | 1 | Software sets this bit after loading a data packet into the TX FIFO. The EP0 bit in the USBTXIS register is also set in this situation. |



Copyright © 2024 Texas Instruments Incorporated

**Table 23-35. USB Control and Status Endpoint 0 Low Register (USBCSRL0) in Host Mode Field Descriptions (continued)**

| Bit | Field | Value | Description |
|---|---|---|---|
| 0 | RXRDY | | Receive Packet Ready. Software must clear this bit after the packet has been read from the FIFO to acknowledge that the data has been read from the FIFO. |
| | | 0 | No receive packet has been received. |
| | | 1 | Indicates that a data packet has been received in the RX FIFO. The EP0 bit in the USBTXIS register is also set in this situation. |

USBCSRL0 in Device mode is shown in Figure 23-34 and described in Table 23-36.

**Figure 23-34. USB Control and Status Endpoint 0 Low Register (USBCSRL0) in Device Mode**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SETENDC | RXRDYC | STALL | SETEND | DATAEND | STALLED | TXRDY | RXRDY |
| W1C-0 | W1C-0 | R/W-0 | R-0 | R/W-0 | R/W-0 | R/W-0 | R-0 |

LEGEND: R/W = Read/Write; R = Read only; W = Write only; -*n* = value after reset

**Table 23-36. USB Control and Status Endpoint 0 Low Register (USBCSRL0) in Device Mode Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7 | SETENDC | | Setup End Clear |
| | | 0 | No effect |
| | | 1 | Writing a 1 to this bit clears the SETEND bit. |
| 6 | RXRDYC | | RXRDY Clear |
| | | 0 | No effect |
| | | 1 | Writing a 1 to this bit clears the RXRDY bit. |
| 5 | STALL | | Send Stall. |
| | | 0 | No effect |
| | | 1 | Terminates the current transaction and transmits the STALL handshake. |
| | | | This bit is cleared automatically after the STALL handshake is transmitted. |
| 4 | SETEND | | Setup end. |
| | | 0 | A control transaction has not ended or ended after the DATAEND bit was set. |
| | | 1 | A control transaction has ended before the DATAEND bit has been set. The EP0 bit in the USBTXIS register is also set in this situation. |
| | | | This bit is cleared by writing a 1 to the SETENDC bit. |
| 3 | DATAEND | | Data end. |
| | | 0 | No effect |
| | | 1 | Set this bit in the following situations:<br>• When setting TXRDY for the last data packet<br>• When clearing RXRDY after unloading the last data packet<br>• When setting TXRDY for a zero-length data packet<br>This bit is cleared automatically. |
| 2 | STALLED | | Endpoint Stalled. Software must clear this bit. |
| | | 0 | A STALL handshake has not been transmitted. |
| | | 1 | A STALL handshake has been transmitted. |



Copyright © 2024 Texas Instruments Incorporated

**Table 23-36. USB Control and Status Endpoint 0 Low Register (USBCSRL0) in Device Mode Field Descriptions (continued)**

| Bit | Field | Value | Description |
|---|---|---|---|
| 1 | TXRDY | | Transmit Packet Ready. If both the TXRDY and SETUP bits are set, a setup packet is sent. If just TXRDY is set, an OUT packet is sent. |
| | | 0 | No transmit packet is ready. |
| | | 1 | Software sets this bit after loading an IN data packet into the TX FIFO. The EP0 bit in the USBTXIS register is also set in this situation. |
| 0 | RXRDY | | Receive Packet Ready. |
| | | 0 | No receive packet has been received. |
| | | 1 | A data packet has been received. The EP0 bit in the USBTXIS register is also set in this situation. |
| | | | This bit is cleared by writing a 1 to the RXRDYC bit. |



Copyright © 2024 Texas Instruments Incorporated

### 23.6.3.28 USB Control and Status Endpoint 0 High Register (USBCSRH0), offset 0x103

The USB control and status endpoint 0 high 8-bit register (USBCSRH0) provides control and status bits for endpoint 0.

**Mode(s):** Host Device

USBCSRH0 in Host mode is shown in Figure 23-35 and described in Table 23-37.

**Figure 23-35. USB Control and Status Endpoint 0 High Register (USBCSRH0) in Host Mode**

| 7 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|
| Reserved | | DTWE | DT | FLUSH |
| R-0 | | R/W-0 | R/W-0 | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

**Table 23-37. USB Control and Status Endpoint 0 High Register (USBCSRH0) in Host Mode Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7-3 | Reserved | 0 | Reserved |
| 2 | DTWE | | Data Toggle Write Enable. This bit is automatically cleared once the new value is written. |
| | | 0 | The DT bit cannot be written. |
| | | 1 | Enables the current state of the endpoint 0 data toggle to be written (see DT bit). |
| 1 | DT | | Data Toggle. When read, this bit indicates the current state of the endpoint 0 data toggle.<br>If DTWE is set, this bit may be written with the required setting of the data toggle. If DTWE is Low, this bit cannot be written. Care should be taken when writing to this bit as it should only be changed to RESET USB endpoint 0. |
| 0 | FLUSH | | Flush FIFO. This bit is automatically cleared after the flush is performed. |
| | | 0 | No effect |
| | | 1 | Flushes the next packet to be transmitted/read from the endpoint 0 FIFO. The FIFO pointer is reset and the TXRDY/RXRDY bit is cleared.<br>**Note:** This bit should only be set when TXRDY/RXRDY is set. At other times, it may cause data to be corrupted. |



Copyright © 2024 Texas Instruments Incorporated

USBCSRH0 in Device mode is shown in Figure 23-36 and described in Table 23-38.

**Figure 23-36. USB Control and Status Endpoint 0 High Register (USBCSRH0) in Device Mode**

| 7 | 1 | 0 |
|---|---|---|
| Reserved | | FLUSH |
| R-0 | | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

**Table 23-38. USB Control and Status Endpoint 0 High Register (USBCSRH0) in Device Mode Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7-1 | Reserved | 0 | Reserved |
| 0 | FLUSH | | Flush FIFO. This bit is automatically cleared after the flush is performed. |
| | | 0 | No effect |
| | | 1 | Flushes the next packet to be transmitted/read from the endpoint 0 FIFO. The FIFO pointer is reset and the TXRDY/RXRDY bit is cleared. |
| | | | **Note:** This bit should only be set when TXRDY/RXRDY is set. At other times, it may cause data to be corrupted. |



Copyright © 2024 Texas Instruments Incorporated

#### 23.6.3.29 USB Receive Byte Count Endpoint 0 Register (USBCOUNT0), offset 0x108

The USB receive byte count endpoint 0 8-bit read-only register (USBCOUNT0) indicates the number of received data bytes in the endpoint 0 FIFO. The value returned changes as the contents of the FIFO change and is only valid while the RXRDY bit is set.

**Mode(s):** Host Device

USBCOUNT0 is shown in Figure 23-37 and described in Table 23-39 .

**Figure 23-37. USB Receive Byte Count Endpoint 0 Register (USBCOUNT0)**

| 7 | 6 | 0 |
|---|---|---|
| Reserved | COUNT | |
| R-0 | R/W-0 | |

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

**Table 23-39. USB Receive Byte Count Endpoint 0 Register (USBCOUNT0) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7 | Reserved | 0 | Reserved |
| 6-0 | COUNT | 0 | FIFO Count. COUNT is a read-only value that indicates the number of received data bytes in the endpoint 0 FIFO. |

#### 23.6.3.30 USB Type Endpoint 0 Register (USBTYPE0), offset 0x10A

The USB type endpoint 0 8-bit register (USBTYPE0) must be written with the operating speed of the targeted Device being communicated with using endpoint 0.

**Mode(s):** Host

USBTYPE0 is shown in Figure 23-38 and described in Table 23-40.

**Figure 23-38. USB Type Endpoint 0 Register (USBTYPE0)**

| 7 | 6 | 5 | 0 |
|---|---|---|---|
| SPEED | | Reserved | |
| R/W-0 | | R-0 | |

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

**Table 23-40. USB Type Endpoint 0 Register (USBTYPE0) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7-6 | SPEED | 0 | Operating Speed specifies the operating speed of the target Device. If selected, the target is assumed to have the same connection speed as the USB controller. |
| | | 0-1h | Reserved |
| | | 2h | Full |
| | | 3h | Low |
| 5-0 | Reserved | 0 | Reserved |



Copyright © 2024 Texas Instruments Incorporated

#### 23.6.3.31 USB NAK Limit Register (USBNAKLMT), offset 0x10B

The USB NAK limit 8-bit read-only register (USBNAKLMT) sets the number of frames after which endpoint 0 should time out on receiving a stream of NAK responses. (Equivalent settings for other endpoints can be made through their USBTXINTERVAL[*n*] and USBRXINTERVAL[*n*] registers.)

The number of frames selected is 2(m-1) (where m is the value set in the register, with valid values of 2–16). If the Host receives NAK responses from the target for more frames than the number represented by the limit set in this register, the endpoint is halted.

**Note:** A value of 0 or 1 disables the NAK timeout function.

**Mode(s):** Host

USBNAKLMT is shown in Figure 23-39 and described in Table 23-41.

**Figure 23-39. USB NAK Limit Register (USBNAKLMT)**

| 7 | 5 | 4 | 0 |
|---|---|---|---|
| Reserved | | NAKLMT | |
| R-0 | | R/W-0 | |

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

**Table 23-41. USB NAK Limit Register (USBNAKLMT) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7-5 | Reserved | 0 | Reserved |
| 4-0 | NAKLMT | 0 | EP0 NAK Limit specifies the number of frames after receiving a stream of NAK responses. |



Copyright © 2024 Texas Instruments Incorporated

#### 23.6.3.32 USB Maximum Transmit Data Endpoint n Registers (USBTXMAXP[1]-USBTXMAXP[*n*])

The USB maximum transmit data endpoint *n* 16-bit registers (USBTXMAXP[*n*]) define the maximum amount of data that can be transferred through the selected transmit endpoint in a single operation.

Bits 10:0 define (in bytes) the maximum payload transmitted in a single transaction. The value set can be up to 1024 bytes but is subject to the constraints placed by the USB Specification on packet sizes for bulk and interrupt transfers in full-speed operation.

The total amount of data represented by the value written to this register must not exceed the FIFO size for the transmit endpoint, and must not exceed half the FIFO size if double-buffering is required.

If this register is changed after packets have been sent from the endpoint, the transmit endpoint FIFO must be completely flushed (using the FLUSH bit in USBTXCSRLn) after writing the new value to this register.

**Note:** USBTXMAXP[*n*] must be set to an even number of bytes for proper interrupt generation in DMA Basic Mode.

**Mode(s):** Host Device

The USBTXMAXP[*n*] registers are shown in Figure 23-40 and described in Table 23-42.

**Figure 23-40. USB Maximum Transmit Data Endpoint *n* Registers (USBTXMAXP[*n*])**

| 15 | 11 | 10 | 0 |
|---|---|---|---|
| Reserved | | MAXLOAD | |
| R-0 | | R/W-000 | |

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

**Table 23-42. USB Maximum Transmit Data Endpoint *n* Registers (USBTXMAXP[*n*]) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 15-11 | Reserved | 0 | Reserved |
| 10-0 | MAXLOAD | | Maximum Payload specifies the maximum payload in bytes per transaction. |



Copyright © 2024 Texas Instruments Incorporated

#### 23.6.3.33 USB Transmit Control and Status Endpoint n Low Register (USBTXCSRL[1]-USBTXCSRL[*n*])

The USB transmit control and status endpoint *n* low 8-bit registers (USBTXCSRL[*n*]) provide control and status bits for transfers through the currently selected transmit endpoint.

**Mode(s):** Host Device

The USBTXCSRL[*n*] registers in Host Mode are shown in Figure 23-41 and described in Table 23-43.

**Figure 23-41. USB Transmit Control and Status Endpoint n Low Register (USBTXCSRL[*n*]) in Host Mode**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| NAKTO | CLRDT | STALLED | SETUP | FLUSH | ERROR | FIFONE | TXRDY |
| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |

LEGEND: R/W = Read/Write; -*n* = value after reset

**Table 23-43. USB Transmit Control and Status Endpoint n Low Register (USBTXCSRL[*n*]) in Host Mode Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7 | NAKTO | | NAK Timeout. Software must clear this bit to allow the endpoint to continue. |
| | | 0 | No timeout |
| | | 1 | Bulk endpoints only: Indicates that the transmit endpoint is halted following the receipt of NAK responses for longer than the time set by the NAKLMT field in the USBTXINTERVAL[*n*] register. |
| 6 | CLRDT | | Clear DataToggle |
| | | 0 | No effect |
| | | 1 | Writing a 1 to this bit clears the DT bit in the USBTXCSRH[*n*] register. |
| 5 | STALLED | | Endpoint Stalled. Software must clear this bit. |
| | | 0 | A STALL handshake has not been received |
| | | 1 | Indicates that a STALL handshake has been received. When this bit is set, any DMA request that is in progress is stopped, the FIFO is completely flushed, and the TXRDY bit is cleared. |
| 4 | SETUP | | Setup Packet. |
| | | 0 | No SETUP token is sent. |
| | | 1 | Sends a SETUP token instead of an OUT token for the transaction. This bit should be set at the same time as the TXRDY bit is set. |
| | | | **Note:** Setting this bit also clears the DT bit in the USBTXCSRH[*n*] register. |
| 3 | FLUSH | | Flush FIFO. This bit can be set simultaneously with the TXRDY bit to abort the packet that is currently being loaded into the FIFO. Note that if the FIFO is double-buffered, FLUSH may have to be set twice to completely clear the FIFO. |
| | | 0 | No effect |
| | | 1 | Flushes the latest packet from the endpoint transmit FIFO. The FIFO pointer is reset and the TXRDY bit is cleared. The EPn bit in the USBTXIS register is also set in this situation. |
| | | | **Note:** This bit should only be set when the TXRDY bit is set. At other times, it may cause data to be corrupted. |
| 2 | ERROR | | Error. Software must clear this bit. |
| | | 0 | No error |
| | | 1 | Three attempts have been made to send a packet and no handshake packet has been received. The TXRDY bit is cleared, the EPn bit in the USBTXIS register is set, and the FIFO is completely flushed in this situation. |
| | | | **Note:** This bit is valid only when the endpoint is operating in Bulk or Interrupt mode. |
| 1 | FIFONE | | FIFO Not Empty |
| | | 0 | The FIFO is empty |
| | | 1 | At least one packet is in the transmit FIFO. |



Copyright © 2024 Texas Instruments Incorporated

**Table 23-43. USB Transmit Control and Status Endpoint n Low Register (USBTXCSRL[*n*]) in Host Mode Field Descriptions (continued)**

| Bit | Field | Value | Description |
|---|---|---|---|
| 0 | TXRDY | | Transmit Packet Ready.<br>This bit is cleared automatically when a data packet has been transmitted. The EP*n* bit in the USBTXIS register is also set at this point. TXRDY is also automatically cleared prior to loading a second packet into a double-buffered FIFO. |
| | | 0 | No transmit packet is ready. |
| | | 1 | Software sets this bit after loading a data packet into the TX FIFO. |

The USBTXCSRL[*n*] registers in Device Mode are shown in Figure 23-42 and described in Table 23-44.

**Figure 23-42. USB Transmit Control and Status Endpoint n Low Register (USBTXCSRL[*n*]) in Device Mode**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | CLRDT | STALLED | STALL | FLUSH | UNDRN | FIFONE | TXRDY |
| R-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

**Table 23-44. USB Transmit Control and Status Endpoint n Low Register (USBTXCSRL[*n*]) in Device Mode Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7 | Reserved | 0 | Reserved |
| 6 | CLRDT | | Clear Data Toggle |
| | | 0 | No effect |
| | | 1 | Writing a 1 to this bit clears the DT bit in the USBTXCSRH[*n*] register. |
| 5 | STALLED | | Endpoint Stalled. Software must clear this bit. |
| | | 0 | A STALL handshake has not been transmitted. |
| | | 1 | A STALL handshake has been transmitted. The FIFO is flushed and the TXRDY bit is cleared. |
| 4 | STALL | | Send Stall. Software clears this bit to terminate the STALL condition. |
| | | 0 | No effect |
| | | 1 | Issues a STALL handshake to an IN token. |
| 3 | FLUSH | | Flush FIFO. This bit may be set simultaneously with the TXRDY bit to abort the packet that is currently being loaded into the FIFO. Note that if the FIFO is double-buffered, FLUSH may have to be set twice to completely clear the FIFO.<br>**Note:** This bit should only be set when the TXRDY bit is set. At other times, it may cause data to be corrupted. |
| | | 0 | No effect |
| | | 1 | Flushes the latest packet from the endpoint transmit FIFO. The FIFO pointer is reset and the TXRDY bit is cleared. The EPn bit in the USBTXIS register is also set in this situation.<br>This bit is cleared automatically. |
| 2 | UNDRN | | Underrun. Software must clear this bit. |
| | | 0 | No underrun |
| | | 1 | An IN token has been received when TXRDY is not set. |
| 1 | FIFONE | | FIFO Not Empty |
| | | 0 | The FIFO is empty. |
| | | 1 | At least one packet is in the transmit FIFO. |



Copyright © 2024 Texas Instruments Incorporated

**Table 23-44. USB Transmit Control and Status Endpoint n Low Register (USBTXCSRL[*n*]) in Device Mode Field Descriptions (continued)**

| Bit | Field | Value | Description |
|---|---|---|---|
| 0 | TXRDY | | Transmit Packet Ready.<br>This bit is cleared automatically when a data packet has been transmitted. The EPn bit in the USBTXIS register is also set at this point. TXRDY is also automatically cleared prior to loading a second packet into a double-buffered FIFO. |
| | | 0 | No transmit packet is ready. |
| | | 1 | Software sets this bit after loading a data packet into the TX FIFO.<br>This bit is cleared by writing a 1 to the RXRDYC bit. |



Copyright © 2024 Texas Instruments Incorporated

#### 23.6.3.34 USB Transmit Control and Status Endpoint n High Register (USBTXCSRH[1]-USBTXCSRH[*n*])

The USB transmit control and status endpoint *n* high 8-bit registers (USBTXCSRH[*n*]) provide additional control for transfers through the currently selected transmit endpoint.

**Mode(s):** Host Device

The USBTXCSRH[*n*] registers in Host Mode are shown in Figure 23-43 and described in Table 23-45.

**Figure 23-43. USB Transmit Control and Status Endpoint n High Register (USBTXCSRH[*n*]) in Host Mode**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| AUTOSET | Reserved | MODE | DMAEN | FDT | DMAMOD | DTWE | DT |
| R/W-0 | R-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |

LEGEND: R/W = Read/Write; -*n* = value after reset

**Table 23-45. USB Transmit Control and Status Endpoint n High Register (USBTXCSRH[*n*]) in Host Mode Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7 | AUTOSET | | Auto Set |
| | | 0 | The TXRDY bit must be set manually. |
| | | 1 | Enables the TXRDY bit to be automatically set when data of the maximum packet size (value in USBTXMAXP[*n*]) is loaded into the transmit FIFO. If a packet of less than the maximum packet size is loaded, then the TXRDY bit must be set manually. |
| 6 | Reserved | 0 | Reserved. Any writes to these bit(s) must always have a value of 0. |
| 5 | MODE | | Mode<br>**Note:** This bit only has an effect when the same endpoint FIFO is used for both transmit and receive transactions. |
| | | 0 | Enables the endpoint direction as RX. |
| | | 1 | Enables the endpoint direction as TX. |
| 4 | DMAEN | | DMA Request Enable<br>**Note:** Three TX and three /RX endpoints can be connected to the DMA module. If this bit is set for a particular endpoint, the DMAATX, DMABTX, or DMACTX field in the USB DMA Select (USBDMASEL) register must be programmed correspondingly. |
| | | 0 | Disables the DMA request for the transmit endpoint. |
| | | 1 | Enables the DMA request for the transmit endpoint. |
| 3 | FDT | | Force Data Toggle |
| | | 0 | No effect |
| | | 1 | Forces the endpoint DT bit to switch and the data packet to be cleared from the FIFO, regardless of whether an ACK was received.<br>**Note:** This bit should only be set when the TXRDY bit is set. At other times, it may cause data to be corrupted. |
| 2 | DMAMOD | | DMA Request Mode<br>**Note:** This bit must not be cleared either before or in the same cycle as the above DMAEN bit is cleared. |
| | | 0 | An interrupt is generated after every DMA packet transfer. |
| | | 1 | An interrupt is generated only after the entire DMA transfer is complete.<br>**Note:** This bit is valid only when the endpoint is operating in Bulk or Interrupt mode. |
| 1 | DTWE | | Data Toggle Write Enable. This bit is automatically cleared once the new value is written. |
| | | 0 | The DT bit cannot be written. |
| | | 1 | Enables the current state of the transmit endpoint data to be written (see DT bit). |



Copyright © 2024 Texas Instruments Incorporated

**Table 23-45. USB Transmit Control and Status Endpoint n High Register (USBTXCSRH[*n*]) in Host Mode Field Descriptions (continued)**

| Bit | Field | Value | Description |
|---|---|---|---|
| 0 | DT | | Data Toggle. When read, this bit indicates the current state of the transmit endpoint data toggle.<br>If DTWE is High, this bit can be written with the required setting of the data toggle. If DTWE is Low, any value written to this bit is ignored. Care should be taken when writing to this bit as it should only be changed to RESET the transmit endpoint. |

The USBTXCSRH[*n*] registers in Device Mode are shown in Figure 23-44 and described in Table 23-46.

**Figure 23-44. USB Transmit Control and Status Endpoint n High Register (USBTXCSRH[*n*]) in Device Mode**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| AUTOSET | Reserved | MODE | DMAEN | FDT | DMAMOD | Reserved | |
| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R-0 | |

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

**Table 23-46. USB Transmit Control and Status Endpoint n High Register (USBTXCSRH[*n*]) in Device Mode Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7 | AUTOSET | | Auto Set |
| | | 0 | The TXRDY bit must be set manually. |
| | | 1 | Enables the TXRDY bit to be automatically set when data of the maximum packet size (value in USBTXMAXP[*n*]) is loaded into the transmit FIFO. If a packet of less than the maximum packet size is loaded, then the TXRDY bit must be set manually. |
| 6 | Reserved | | Reserved. Should always have a value of 0. |
| 5 | MODE | | Mode<br>**Note:** This bit only has an effect when the same endpoint FIFO is used for both transmit and receive transactions. |
| | | 0 | Enables the endpoint direction as RX. |
| | | 1 | Enables the endpoint direction as TX. |
| 4 | DMAEN | | DMA Request Enable<br>**Note:** Three TX and three /RX endpoints can be connected to the DMA module. If this bit is set for a particular endpoint, the DMAATX, DMABTX, or DMACTX field in the USB DMA Select (USBDMASEL) register must be programmed correspondingly. |
| | | 0 | Disables the DMA request for the transmit endpoint. |
| | | 1 | Enables the DMA request for the transmit endpoint. |
| 3 | FDT | | Force Data Toggle |
| | | 0 | No effect |
| | | 1 | Forces the endpoint DT bit to switch and the data packet to be cleared from the FIFO, regardless of whether an ACK was received. |
| 2 | DMAMOD | | DMA Request Mode<br>**Note:** This bit must not be cleared either before or in the same cycle as the above DMAEN bit is cleared. |
| | | 0 | An interrupt is generated after every DMA packet transfer. |
| | | 1 | An interrupt is generated only after the entire DMA transfer is complete. |
| 0 | Reserved | 0 | Reserved |



Copyright © 2024 Texas Instruments Incorporated

#### 23.6.3.35 USB Maximum Receive Data Endpoint n Registers (USBRXMAXP[1]-USBRXMAXP[*n*])

The USB maximum receive data endpoint *n* 16-bit registers (USBRXMAXP[*n*]) define the maximum amount of data that can be transferred through the selected receive endpoint in a single operation.

Bits 10:0 define (in bytes) the maximum payload transmitted in a single transaction. The value set can be up to 1024 bytes but is subject to the constraints placed by the USB Specification on packet sizes for bulk and interrupt transfers in full-speed operation.

The total amount of data represented by the value written to this register must not exceed the FIFO size for the transmit endpoint, and must not exceed half the FIFO size if double-buffering is required.

**Note:** USBRXMAXP[*n*] must be set to an even number of bytes for proper interrupt generation in DMA Basic Mode.

**Mode(s):** Host Device

The USBRXMAXP[*n*] registers are shown in Figure 23-45 and described in Table 23-47.

**Figure 23-45. USB Maximum Receive Data Endpoint *n* Registers (USBRXMAXP[*n*])**

| 15 | 11 | 10 | 0 |
|---|---|---|---|
| Reserved | | MAXLOAD | |
| R-0 | | R/W-000 | |

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

**Table 23-47. USB Maximum Receive Data Endpoint *n* Registers (USBRXMAXP[*n*]) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 15-11 | Reserved | 0 | Reserved |
| 10-0 | MAXLOAD | | Maximum Payload specifies the maximum payload in bytes per transaction. |



Copyright © 2024 Texas Instruments Incorporated

#### 23.6.3.36 USB Receive Control and Status Endpoint n Low Register (USBRXCSRL[1]-USBRXCSRL[*n*])

The USB receive control and status endpoint *n* low 8-bit register (USBRXCSRL[*n*]) provides control and status bits for transfers through the currently selected receive endpoint.

**Mode(s):** Host Device

The USBCSRL[*n*] registers in Host mode are shown in Figure 23-46 and described in Table 23-48.

**Figure 23-46. USB Receive Control and Status Endpoint n Low Register (USBRXCSRL[*n*]) in Host Mode**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| CLRDT | STALLED | REQPKT | FLUSH | DATAERR / NAKTO | ERROR | FULL | RXRDY |
| W1C-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |

LEGEND: R/W = Read/Write; W = Write only; -*n* = value after reset

**Table 23-48. USB Receive Control and Status Endpoint n Low Register (USBRXCSRL[*n*]) in Host Mode Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7 | NAKTO | | Clear Data Toggle. |
| | | 0 | No effect |
| | | 1 | Writing a 1 to this bit clears the DT bit in the USBRXCSRH[*n*] register. |
| 6 | STALLED | | Endpoint Stalled. Software must clear this bit. |
| | | 0 | No handshake has been received. |
| | | 1 | A STALL handshake has been received. The EP*n* bit in the USBRXIS register is also set. |
| 5 | REQPKT | | Request Packet. This bit is cleared when the RXRDY bit is set. |
| | | 0 | No request |
| | | 1 | Requests an IN transaction. |
| 4 | FLUSH | | Flush FIFO. If the FIFO is double-buffered, FLUSH may have to be set twice to completely clear the FIFO.<br>**Note:**This bit should only be set when the RXRDY bit is set. At other times, it may cause data to be corrupted. |
| | | 0 | No effect |
| | | 1 | Flushes the next packet to be read from the endpoint receive FIFO. The FIFO pointer is reset and the RXRDY bit is cleared. |
| 3 | DATAERR / NAKTO | | Data Error / NAK Timeout |
| | | 0 | Normal operation |
| | | 1 | Bulk endpoints only: Indicates that the receive endpoint is halted following the receipt of NAK responses for longer than the time set by the NAKLMT field in the USBRXINTERVAL[*n*] register. Software must clear this bit to allow the endpoint to continue. |
| 2 | ERROR | | Error. Software must clear this bit.<br>**Note:** This bit is only valid when the receive endpoint is operating in Bulk or Interrupt mode. |
| | | 0 | No error |
| | | 1 | Three attempts have been made to receive a packet and no data packet has been received. The EP*n* bit in the USBRXIS register is set in this situation. |
| 1 | FULL | | FIFO Full |
| | | 0 | The receive FIFO is not full. |
| | | 1 | No more packets can be loaded into the receive FIFO. |



Copyright © 2024 Texas Instruments Incorporated

**Table 23-48. USB Receive Control and Status Endpoint n Low Register (USBRXCSRL[*n*]) in Host Mode Field Descriptions (continued)**

| Bit | Field | Value | Description |
|---|---|---|---|
| 0 | RXRDY | | Receive Packet Ready.<br>If the AUTOCLR bit in the USBRXCSRH[*n*] register is set, then the this bit is automatically cleared when a packet of USBRXMAXP[*n*] bytes has been unloaded from the receive FIFO. If the AUTOCLR bit is clear, or if packets of less than the maximum packet size are unloaded, then software must clear this bit manually when the packet has been unloaded from the receive FIFO. |
| | | 0 | No data packet has been received. |
| | | 1 | Indicates that a data packet has been received. The EP*n* bit in the USBTXIS register is also set in this situation. |

USBCSRL0 in Device mode is shown in Figure 23-47 and described in Table 23-49.

**Figure 23-47. USB Receive Control and Status Endpoint n Low Register (USBRXCSRL[*n*]) in Device Mode**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| CLRDT | STALLED | STALL | FLUSH | Reserved | | FULL | RXRDY |
| W1C-0 | W1C-0 | R/W-0 | R-0 | R/W-0 | R/W-0 | R/W-0 | R-0 |

LEGEND: R/W = Read/Write; R = Read only; W = Write only; -*n* = value after reset

**Table 23-49. USB Receive Control and Status Endpoint n Low Register (USBRX CSRL[*n*]) in Device Mode Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7 | CLRDT | | Clear Data Toggle |
| | | 0 | No effect |
| | | 1 | Writing a 1 to this bit clears the DT bit in the USBRXCSRH[*n*] register. |
| 6 | STALLED | | Endpoint Stalled. Software must clear this bit. |
| | | 0 | A STALL handshake has been transmitted. |
| | | 1 | A STALL handshake has been transmitted. |
| 5 | STALL | | Send Stall. Software must clear this bit to terminate the STALL condition. |
| | | 0 | No effect |
| | | 1 | Issues a STALL handshake. |
| 4 | FLUSH | | Flush FIFO. The CPU writes a 1 to this bit to flush the next packet to be read from the endpoint receive FIFO. The FIFO pointer is reset and the RXRDY bit is cleared. Note that if the FIFO is double-buffered, FLUSH may have to be set twice to completely clear the FIFO. |
| | | 0 | No effect |
| | | 1 | Flushes the next packet from the endpoint receive FIFO. The FIFO pointer is reset and the RXRDY bit is cleared.<br>**Note:** This bit should only be set when the RXRDY bit is set. At other times, it may cause data to be corrupted. |
| 3-2 | Reserved | | Reserved |
| 1 | FULL | | FIFO Full |
| | | 0 | The receive FIFO is not full. |
| | | 1 | No more packets can be loaded into the receive FIFO. |



Copyright © 2024 Texas Instruments Incorporated

**Table 23-49. USB Receive Control and Status Endpoint n Low Register (USBRX CSRL[*n*]) in Device Mode Field Descriptions (continued)**

| Bit | Field | Value | Description |
|---|---|---|---|
| 0 | RXRDY | | Receive Packet Ready.<br>If the AUTOCLR bit in the USBRXCSRH[*n*] register is set, then the this bit is automatically cleared when a packet of USBRXMAXP[*n*] bytes has been unloaded from the receive FIFO. If the AUTOCLR bit is clear, or if packets of less than the maximum packet size are unloaded, then software must clear this bit manually when the packet has been unloaded from the receive FIFO. |
| | | 0 | No data packet has been received. |
| | | 1 | A data packet has been received. The EP*n* bit in the USBTXIS register is also set in this situation.<br>This bit is cleared by writing a 1 to the RXRDYC bit. |



Copyright © 2024 Texas Instruments Incorporated

### 23.6.3.37 USB Receive Control and Status Endpoint n High Register (USBRXCSRH[1]-USBRXCSRH[*n*])

The USB receive control and status endpoint *n* high 8-bit register (USBRXCSRL[*n*]) provides additional control and status bits for transfers through the currently selected receive endpoint.

**Mode(s):** Host Device

The USBCSRH[*n*] registers in OTG A/Host mode are shown in Figure 23-48 and described in Table 23-50.

**Figure 23-48. USB Receive Control and Status Endpoint n High Register (USBRXCSRH[*n*]) in Host Mode**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| AUTOCL | AUTORQ | DMAEN | Reserved | DMAMOD | DTWE | DT | Reserved |
| W1C-0 | R/W-0 | R/W-0 | R-0 | R/W-0 | R-0 | R-0 | R-0 |

LEGEND: R/W = Read/Write; -*n* = value after reset

**Table 23-50. USB Receive Control and Status Endpoint n High Register (USBRXCSRH[*n*]) in Host Mode Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7 | AUTOCL | | Auto Clear |
| | | 0 | No effect |
| | | 1 | Enables the RXRDY bit to be automatically cleared when a packet of USBRXMAXP[*n*] bytes has been unloaded from the receive FIFO. When packets of less than the maximum packet size are unloaded, RXRDY must be cleared manually. Care must be taken when using DMA to unload the receive FIFO as data is read from the receive FIFO in 4-byte chunks regardless of the value of the MAXLOAD field in the USBRXMAXP[*n*] register, see Section 23.2.3. |
| 6 | AUTORQ | | Auto Request<br>**Note:** This bit is automatically cleared when a short packet is received. |
| | | 0 | No effect |
| | | 1 | Enables the REQPKT bit to be automatically set when the RXRDY bit is cleared. |
| 5 | DMAEN | | DMA Request Enable<br>**Note:** Three TX and three RX endpoints can be connected to the DMA module. If this bit is set for a particular endpoint, the DMAARX, DMABRX, or DMACRX field in the USB DMA Select (USBDMASEL) register must be programmed correspondingly. |
| | | 0 | Disables the DMA request for the receive endpoint. |
| | | 1 | Enables the DMA request for the receive endpoint. |
| 4 | Reserved | | Reserved |
| 3 | DMAMOD | | DMAMOD<br>**Note:** This bit must not be cleared either before or in the same cycle as the above DMAEN bit is cleared. |
| | | 0 | An interrupt is generated after every DMA packet transfer. |
| | | 1 | An interrupt is generated only after the entire DMA transfer is complete. |
| 2 | DTWE | | Data Toggle Write Enable. This bit is automatically cleared once the new value is written. |
| | | 0 | The DT bit cannot be written. |
| | | 1 | Enables the current state of the receive endpoint data to be written (see DT bit). |
| 1 | DT | | Data Toggle. When read, this bit indicates the current state of the receive data toggle.<br>If DTWE is High, this bit may be written with the required setting of the data toggle. If DTWE is Low, any value written to this bit is ignored. Care should be taken when writing to this bit as it should only be changed to RESET the receive endpoint. |
| 0 | Reserved | 0 | Reserved |



Copyright © 2024 Texas Instruments Incorporated

The USBCSRH[*n*] registers in Device mode are shown in Figure 23-49 and described in Table 23-51.

**Figure 23-49. USB Receive Control and Status Endpoint n High Register (USBRXCSRH[*n*]) in Device Mode**

| 7 | 6 | 5 | 4 | 3 | 2 | | 0 |
|---|---|---|---|---|---|---|---|
| AUTOCL | Reserved | DMAEN | DISNYET / PIDERR | DMAMOD | | Reserved | |
| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | | R-0 | |

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

**Table 23-51. USB Receive Control and Status Endpoint n High Register (USBRXCSRH[*n*]) in Device Mode Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7 | AUTOCL | | Auto Clear |
| | | 0 | No effect |
| | | 1 | Enables the RXRDY bit to be automatically cleared when a packet of USBRXMAXP[*n*] bytes has been unloaded from the receive FIFO. When packets of less than the maximum packet size are unloaded, RXRDY must be cleared manually. Care must be taken when using DMA to unload the receive FIFO as data is read from the receive FIFO in 4-byte chunks regardless of the value of the MAXLOAD field in the USBRXMAXP[*n*] register, see Section 23.2.3. |
| 6 | Reserved | | Reserved |
| 5 | DMAEN | | DMA Request Enable<br>**Note:** Three TX and three RX endpoints can be connected to the DMA module. If this bit is set for a particular endpoint, the DMAARX, DMABRX, or DMACRX field in the USB DMA Select (USBDMASEL) register must be programmed correspondingly. |
| | | 0 | Disables the DMA request for the receive endpoint. |
| | | 1 | Enables the DMA request for the receive endpoint. |
| 4 | DISNYET/ PIDERR | | Disable NYET / PID Error |
| | | 0 | No effect |
| | | 1 | For bulk or interrupt transactions: Disables the sending of NYET handshakes. When this bit is set, all successfully received packets are acknowledged, including at the point at which the FIFO becomes full. |
| 3 | DMAMOD | | DMA Request Mode<br>**Note:** This bit must not be cleared either before or in the same cycle as the above DMAEN bit is cleared. |
| | | 0 | An interrupt is generated after every DMA packet transfer. |
| | | 1 | An interrupt is generated only after the entire DMA transfer is complete. |
| 0 | Reserved | 0 | Reserved |



Copyright © 2024 Texas Instruments Incorporated

#### 23.6.3.38 USB Receive Byte Count Endpoint n Register (USBRXCOUNT[1]-USBRXCOUNT[*n*])

The USB receive byte count endpoint *n* 16-bit read-only registers hold the number of data bytes in the packet currently in line to be read from the receive FIFO. If the packet is transmitted as multiple bulk packets, the number given is for the combined packet.

**Note:** The value returned changes as the FIFO is unloaded and is only valid while the RXRDY bit in the USBRXCSRLn register is set.

**Mode(s):** Host Device

The USBRXCOUNT[*n*] registers are shown in Figure 23-50 and described in Table 23-52.

**Figure 23-50. USB Receive Byte Count Endpoint n Register (USBRXCOUNT[*n*])**

| 15 13 | 12 0 |
|---|---|
| Reserved | COUNT |
| R-0 | R-0 |

LEGEND: R = Read only; -*n* = value after reset

**Table 23-52. USB Receive Byte Count Endpoint n Register (USBRXCOUNT[*n*]) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 15-13 | Reserved | 0 | Reserved |
| 12-0 | COUNT | | Receive Packet Count indicates the number of bytes in the receive packet. |



Copyright © 2024 Texas Instruments Incorporated

#### 23.6.3.39 USB Host Transmit Configure Type Endpoint n Registers (USBTXTYPE[1]-USBTXTYPE[*n*])

The USB host transmit configure type endpoint *n* 8-bit registers (USBTXTYPE[*n*]) must be written with the endpoint number to be targeted by the endpoint, the transaction protocol to use for the currently selected transmit endpoint, and its operating speed.

**Mode(s):** Host

The USBTXTYPE[*n*] registers are shown in Figure 23-51 and described in Table 23-53.

**Figure 23-51. USB Host Transmit Configure Type Endpoint n Registers (USBTXTYPE[*n*])**

| 7 | 6 | 5 | 4 | 3 | 0 |
|---|---|---|---|---|---|
| SPEED | | PROTO | | TEP | |
| R/W-0 | | R/W-0 | | R/W-0 | |

LEGEND: R/W = Read/Write; -*n* = value after reset

**Table 23-53. USB Host Transmit Configure Type Endpoint n Registers (USBTXTYPE[*n*]) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7-6 | SPEED | | Operating Speed. This bit field specifies the operating speed of the target Device: |
| | | 0h | Default. The target is assumed to be using the same connection speed as the USB controller. |
| | | 1h | Reserved |
| | | 2h | Full |
| | | 3h | Low |
| 5-4 | PROTO | | Protocol. Software must configure this bit field to select the required protocol for the transmit endpoint: |
| | | 0h | Control |
| | | 1h | Reserved |
| | | 2h | Bulk |
| | | 3h | Interrupt |
| 3-0 | TEP | 0 | Target Endpoint Number. Software must configure this value to the endpoint number contained in the transmit endpoint descriptor returned to the USB controller during Device enumeration. |



Copyright © 2024 Texas Instruments Incorporated

#### 23.6.3.40 USB Host Transmit Interval Endpoint n Registers (USBTXINTERVAL[1]-USBTXINTERVAL[*n*])

The USB host transmit interval endpoint *n* 8-bit registers (USBTXINTERVAL[*n*]), for interrupt transfers, define the polling interval for the currently selected transmit endpoint. For bulk endpoints, this register defines the number of frames after which the endpoint should time out on receiving a stream of NAK responses.

The USBTXINTERVAL[*n*] registers values define a number of frames, as follows:

**Table 23-54. USB Host Transmit Interval Endpoint *n* Registers (USBTXINTERVAL[*n*]) Frame Numbers**

| Transfer Type | Speed | Valid Values (m) | Interpretation |
|---|---|---|---|
| Interrupt | Low-speed or Full-speed | 0x01-0xFF | The polling interval is *m* frames. |
| Bulk | Full-speed | 0x02-0x10 | The NAK Limit is $2^{(m-1)}$ frames. A value of 0 or 1 disables the NAK timeout function. |

**Mode(s):** Host

The USBTXINTERVAL[*n*] registers are shown in Figure 23-51 and described in Table 23-53.

**Figure 23-52. USB Host Transmit Interval Endpoint *n* Registers (USBTXINTERVAL[*n*])**

| 7 | 0 |
|---|---|
| TXPOLL / NAKLMT | |
| R/W-0 | |

LEGEND: R/W = Read/Write; -*n* = value after reset

**Table 23-55. USB Host Transmit Interval Endpoint *n* Registers (USBTXINTERVAL[*n*]) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7-0 | TXPOLL / NAKLMT | 0 | TX Polling / NAK Limit The polling interval for interrupt transfers; the NAK limit for bulk transfers. See Table 23-54 for valid entries; other values are reserved. |



Copyright © 2024 Texas Instruments Incorporated

#### 23.6.3.41 USB Host Configure Receive Type Endpoint n Register (USBRXTYPE[1]-USBRXTYPE[*n*])

The USB host configure receive type endpoint *n* 8-bit registers (USBRXTYPE[*n*]) must be written with the endpoint number to be targeted by the endpoint, the transaction protocol to use for the currently selected receive endpoint, and its operating speed.

**Mode(s):** Host

The USBRXTYPE[*n*] registers are shown in Figure 23-53 and described in Table 23-56.

**Figure 23-53. USB Host Configure Receive Type Endpoint n Register (USBRXTYPE[*n*])**

| 7 | 6 | 5 | 4 | 3 | 0 |
|---|---|---|---|---|---|
| SPEED | | PROTO | | TEP | |
| R/W-0 | | R/W-0 | | R/W-0 | |

LEGEND: R/W = Read/Write; -*n* = value after reset

**Table 23-56. USB Host Configure Receive Type Endpoint n Register (USBRXTYPE[*n*]) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7-6 | SPEED | | Operating Speed. This bit field specifies the operating speed of the target Device: |
| | | 0h | Default. The target is assumed to be using the same connection speed as the USB controller. |
| | | 1h | Reserved |
| | | 2h | Full |
| | | 3h | Low |
| 5-4 | PROTO | | Protocol. Software must configure this bit field to select the required protocol for the receive endpoint: |
| | | 0h | Control |
| | | 1h | Reserved |
| | | 2h | Bulk |
| | | 3h | Interrupt |
| 3-0 | TEP | 0 | Target Endpoint Number. Software must configure this value to the endpoint number contained in the transmit endpoint descriptor returned to the USB controller during Device enumeration. |



Copyright © 2024 Texas Instruments Incorporated

#### 23.6.3.42 USB Host Receive Polling Interval Endpoint n Registers (USBRXINTERVAL[1]-USBRXINTERVAL[*n*])

The USB host receive polling interval endpoint *n* 8-bit registers (USBRXINTERVAL[*n*]), for interrupt transfers, define the polling interval for the currently selected transmit endpoint. For bulk endpoints, this register defines the number of frames after which the endpoint should time out on receiving a stream of NAK responses.

The USBRXINTERVAL[*n*] registers values define a number of frames, as follows:

**Table 23-57. USB Host Receive Polling Interval Endpoint *n* Registers (USBRXINTERVAL[*n*]) Frame Numbers**

| Transfer Type | Speed | Valid Values (m) | Interpretation |
|---|---|---|---|
| Interrupt | Low-speed or Full-speed | 0x01-0xFF | The polling interval is *m* frames. |
| Bulk | Full-speed | 0x02-0x10 | The NAK Limit is $2^{(m-1)}$ frames. A value of 0 or 1 disables the NAK timeout function. |

**Mode(s):** Host

The USBRXINTERVAL[*n*] registers are shown in Figure 23-51 and described in Table 23-53.

**Figure 23-54. USB Host Receive Polling Interval Endpoint *n* Registers (USBRXINTERVAL[*n*])**

| 7 | 0 |
|---|---|
| TXPOLL / NAKLMT | |
| R/W-0 | |

LEGEND: R/W = Read/Write; -*n* = value after reset

**Table 23-58. USB Host Receive Polling Interval Endpoint *n* Registers (USBRXINTERVAL[*n*]) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7-0 | TXPOLL / NAKLMT | 0 | TX Polling / NAK Limit The polling interval for interrupt transfers; the NAK limit for bulk transfers. See Table 23-57 for valid entries; other values are reserved. |



Copyright © 2024 Texas Instruments Incorporated

#### 23.6.3.43 USB Request Packet Count in Block Transfer Endpoint n Registers (USBRQPKTCOUNT[1]-USBRQPKTCOUNT[*n*])

The USB receive packet count in block transfer endpoint *n* 16-bit read/writer registers are used in Host mode to specify the number of packets that are to be transferred in a block transfer of one or more bulk packets to receive endpoint *n*. The USB controller uses the value recorded in this register to determine the number of requests to issue where the AUTORQ bit in the USBRXCSRH[*n*] register has been set. For more information about IN transactions as a host, see Section 23.2.2.2.

**Note:** Multiple packets combined into a single bulk packet within the FIFO count as one packet.

**Mode(s):** Host

The USBRQPKTCOUNT[*n*] registers are shown in Figure 23-55 and described in Table 23-59.

**Figure 23-55. USB Request Packet Count in Block Transfer Endpoint *n* Registers (USBRQPKTCOUNT[*n*])**

| 15 13 | 12 0 |
|---|---|
| Reserved | COUNT |
| R-0 | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

**Table 23-59. USB Request Packet Count in Block Transfer Endpoint *n* Registers (USBRQPKTCOUNT[*n*]) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 15-13 | Reserved | 0 | Reserved |
| 12-0 | COUNT | | Block Transfer Packet Count sets the number of packets of the size defined by the MAXLOAD bit field that are to be transferred in a block transfer.<br>**Note:** This is only used in Host mode when AUTORQ is set. The bit has no effect in Device mode or when AUTORQ is not set. |



Copyright © 2024 Texas Instruments Incorporated

#### 23.6.3.44 USB Receive Double Packet Buffer Disable Register (USBRXDPKTBUFDIS), offset 0x340

The USB receive double packet buffer disable 16-bit register (USBRXDPKTBUFDIS) indicates which of the receive endpoints have disabled the double-packet buffer functionality (see *Double-Packet Buffering* in Section 23.2.1.1.1).

**Note:** The USBRXDPKTBUFDIS register is not applicable to the control IN and control OUT endpoints, therefore the EP0 bit does not exist for the USBRXDPKTBUFDIS register.

**Mode(s):** Host Device

USBRXDPKTBUFDIS is shown in Figure 23-56 and described in Table 23-60.

**Figure 23-56. USB Receive Double Packet Buffer Disable Register (USBRXDPKTBUFDIS)**

| 15 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|
| Reserved | | EP3 | EP2 | EP1 | Rsvd |
| R-0 | | R/W-1 | R/W-1 | R/W-1 | R-0 |

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

**Table 23-60. USB Receive Double Packet Buffer Disable Register (USBRXDPKTBUFDIS) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 15-4 | Reserved | | Reserved |
| 3 | EP3 | | EP3 RX Double-Packet Buffer Disable |
| | | 0 | Disables double-packet buffering. |
| | | 1 | Enables double-packet buffering. |
| 2 | EP2 | | EP2 RX Double-Packet Buffer Disable |
| | | 0 | Disables double-packet buffering. |
| | | 1 | Enables double-packet buffering. |
| 1 | EP1 | | EP1 RX Double-Packet Buffer Disable |
| | | 0 | Disables double-packet buffering. |
| | | 1 | Enables double-packet buffering. |
| 0 | Reserved | 0 | Reserved |



Copyright © 2024 Texas Instruments Incorporated

#### 23.6.3.45 USB Transmit Double Packet Buffer Disable Register (USBTXDPKTBUFDIS), offset 0x342

The USB transmit double packet buffer disable 16-bit register (USBTXDPKTBUFDIS) indicates which of the transmit endpoints have disabled the double-packet buffer functionality (see *Double-Packet Buffering* in Section 23.2.1.1.1).

**Note:** The USBTXDPKTBUFDIS register is not applicable to the control IN and control OUT endpoints, therefore the EP0 bit does not exist for the USBTXDPKTBUFDIS register.

**Mode(s):** Host Device

USBTXDPKTBUFDIS is shown in Figure 23-57 and described in Table 23-61.

**Figure 23-57. USB Transmit Double Packet Buffer Disable Register (USBTXDPKTBUFDIS)**

| 15 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|
| Reserved | | EP3 | EP2 | EP1 | Rsvd |
| R-0 | | R/W-1 | R/W-1 | R/W-1 | R-0 |

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

**Table 23-61. USB Transmit Double Packet Buffer Disable Register (USBTXDPKTBUFDIS) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 15-4 | Reserved | | Reserved |
| 3 | EP3 | | EP3 RX Double-Packet Buffer Disable |
| | | 0 | Disables double-packet buffering. |
| | | 1 | Enables double-packet buffering. |
| 2 | EP2 | | EP2 RX Double-Packet Buffer Disable |
| | | 0 | Disables double-packet buffering. |
| | | 1 | Enables double-packet buffering. |
| 1 | EP1 | | EP1 RX Double-Packet Buffer Disable |
| | | 0 | Disables double-packet buffering. |
| | | 1 | Enables double-packet buffering. |
| 0 | Reserved | 0 | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 23.6.3.46 USB External Power Control Register (USBEPC), offset 0x400

The USB external power control 32-bit register (USBEPC) specifies the function of the two-pin external power interface (USB0EPEN and USB0PFLT). The assertion of the power fault input may generate an automatic action, as controlled by the hardware configuration registers. The automatic action is necessary because the fault condition may require a response faster than one provided by firmware.

**Mode(s):** Host Device

USBEPC is shown in Figure 23-58 and described in Table 23-62.

**Figure 23-58. USB External Power Control Register (USBEPC)**

| 31 | 10 | 9 | 8 |
|---|---|---|---|
| Reserved | | PFLTACT | |
| R-0 | | R/W-0 | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | PFLTAEN | PFLTSEN | PFLTEN | Reserved | EPENDE | EPEN | |
| R-0 | R/W-0 | R/W-0 | R/W-0 | R-0 | R/W-0 | R/W-0 | |

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

**Table 23-62. USB External Power Control Register (USBEPC) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-10 | Reserved | 0 | Reserved |
| 9-8 | PFLTACT | | Power Fault Action. This bit field specifies how the USB0EPEN signal is changed when detecting a USB power fault. |
| | | 0h | Unchanged. USB0EPEN is controlled by the combination of the EPEN and EPENDE bits. |
| | | 1h | Tristate. USB0EPEN is undriven (tristate). |
| | | 2h | Low. USB0EPEN is driven Low. |
| | | 3h | High. USB0EPEN is driven High. |
| 7 | Reserved | 0 | Reserved |
| 6 | PFLTAEN | | Power Fault Action Enable. This bit specifies whether a USB power fault triggers any automatic corrective action regarding the driven state of the USB0EPEN signal. |
| | | 0 | Disabled. USB0EPEN is controlled by the combination of the EPEN and EPENDE bits. |
| | | 1 | Enabled. The USB0EPEN output is automatically changed to the state specified by the PFLTACT field. |
| 5 | PFLTSEN | | Power Fault Sense. This bit specifies the logical sense of the USB0PFLT input signal that indicates an error condition.<br>The complementary state is the inactive state. |
| | | 0 | Low Fault. If USB0PFLT is driven Low, the power fault is signaled internally (if enabled by the PFLTEN bit). |
| | | 1 | High Fault. If USB0PFLT is driven High, the power fault is signaled internally (if enabled by the PFLTEN bit). |
| 4 | PFLTEN | | Power Fault Input Enable. This bit specifies whether the USB0PFLT input signal is used in internal logic. |
| | | 0 | Not Used. The USB0PFLT signal is ignored. |
| | | 1 | Used. The USB0PFLT signal is used internally |
| 3 | Reserved | 0 | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 23-62. USB External Power Control Register (USBEPC) Field Descriptions (continued)**

| Bit | Field | Value | Description |
|---|---|---|---|
| 2 | EPENDE | | EPEN Drive Enable. This bit specifies whether the USB0EPEN signal is driven or undriven (tristate). When driven, the signal value is specified by the EPEN field. When not driven, the EPEN field is ignored and the USB0EPEN signal is placed in a high-impedance state.<br>The USB0EPEN signal is undriven at reset because the sense of the external power supply enable is unknown. By adding the high-impedance state, system designers can bias the power supply enable to the disabled state using a large resistor (100 kΩ) and later configure and drive the output signal to enable the power supply. |
| | | 0 | Not Driven. The USB0EPEN signal is high impedance. |
| | | 1 | Driven. The USB0EPEN signal is driven to the logical value specified by the value of the EPEN field. |
| 1-0 | EPEN | | External Power Supply Enable Configuration. This bit field specifies and controls the logical value driven on the USB0EPEN signal. |
| | | 0h | Power Enable Active Low. The USB0EPEN signal is driven Low if the EPENDE bit is set. |
| | | 1h | Power Enable Active High. The USB0EPEN signal is driven High if the EPENDE bit is set. |
| | | 2h | Power Enable High if VBUS Low. The USB0EPEN signal is driven High when the A device is not recognized. |
| | | 3h | Power Enable High if VBUS High. The USB0EPEN signal is driven High when the A device is recognized. |



Copyright © 2024 Texas Instruments Incorporated

### 23.6.3.47 USB External Power Control Raw Interrupt Status Register (USBEPCRIS), offset 0x404

The USB external power control raw interrupt status 32-bit register (USBEPCRIS) specifies the unmasked interrupt status of the two-pin external power interface.

**Mode(s):** Host Device

USBEPCRIS is shown in Figure 23-59 and described in Table 23-63.

**Figure 23-59. USB External Power Control Raw Interrupt Status Register (USBEPCRIS)**

| 31 | 1 | 0 |
|---|---|---|
| Reserved | | PF |
| R-0 | | R-0 |

LEGEND: R = Read only; *-n* = value after reset

**Table 23-63. USB External Power Control Raw Interrupt Status Register (USBEPCRIS) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-1 | Reserved | 0 | Reserved |
| 0 | PF | | USB Power Fault Interrupt Status.<br>This bit is cleared by writing a 1 to the PF bit in the USBEPCISC register. |
| | | 0 | A Power Fault status has been detected. |
| | | 1 | An interrupt has not occurred. |



Copyright © 2024 Texas Instruments Incorporated

#### 23.6.3.48 USB External Power Control Interrupt Mask Register (USBEPCIM), offset 0x408

The USB external power control interrupt mask 32-bit register (USBEPCIM) specifies the interrupt mask of the two-pin external power interface.

**Mode(s):** Host Device

USBEPCIM is shown in Figure 23-60 and described in Table 23-64.

**Figure 23-60. USB External Power Control Interrupt Mask Register (USBEPCIM)**

| 31 | 1 | 0 |
|---|---|---|
| Reserved | | PF |
| R-0 | | R-0 |

LEGEND: R = Read only; *-n* = value after reset

**Table 23-64. USB External Power Control Interrupt Mask Register (USBEPCIM) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-1 | Reserved | 0 | Reserved |
| 0 | PF | | USB Power Fault Interrupt Mask. |
| | | 0 | The raw interrupt signal from a detected power fault is sent to the interrupt controller. |
| | | 1 | A detected power fault does not affect the interrupt status. |



Copyright © 2024 Texas Instruments Incorporated

### 23.6.3.49 USB External Power Control Interrupt Status and Clear Register (USBEPCISC), offset 0x40C

The USB external power control interrupt status and clear 32-bit register (USBEPCISC) specifies the unmasked interrupt status of the two-pin external power interface.

**Mode(s):** Host Device

USBEPCISC is shown in Figure 23-61 and described in Table 23-65.

**Figure 23-61. USB External Power Control Interrupt Status and Clear Register (USBEPCISC)**

| 31 | 1 | 0 |
|---|---|---|
| Reserved | | PF |
| R-0 | | R-0 |

LEGEND: R = Read only; *-n* = value after reset

**Table 23-65. USB External Power Control Interrupt Status and Clear Register (USBEPCISC) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-1 | Reserved | 0 | Reserved. Reset is 0x0000.000. |
| 0 | PF | | USB Power Fault Interrupt Status and Clear.<br>This bit is cleared by writing a 1. Clearing this bit also clears the PF bit in the USBEPCISC register. |
| | | 0 | The PF bits in the USBEPCRIS and USBEPCIM registers are set, providing an interrupt to the interrupt controller. |
| | | 1 | No interrupt has occurred or the interrupt is masked. |



Copyright © 2024 Texas Instruments Incorporated

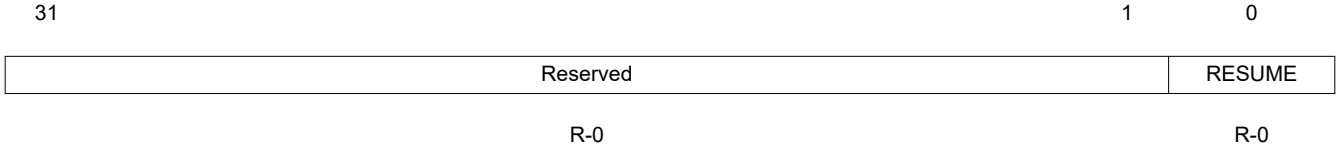
#### 23.6.3.50 USB Device RESUME Raw Interrupt Status Register (USBDRRIS), offset 0x410

The USB device RESUME raw interrupt status register (USBDRRIS) is the raw interrupt status register. On a read, this register gives the current raw status value of the corresponding interrupt prior to masking. A write has no effect.

**Mode(s):** Host Device

USBDRRIS is shown in Figure 23-62 and described in Table 23-66.

**Figure 23-62. USB Device RESUME Raw Interrupt Status Register (USBDRRIS)**




LEGEND: R = Read only; *-n* = value after reset

**Table 23-66. USB Device RESUME Raw Interrupt Status Register (USBDRRIS) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-1 | Reserved | 0 | Reserved. Reset is 0x0000.000. |
| 0 | PF | | RESUME Interrupt Status<br>This bit is cleared by writing a 1 to the RESUME bit in the USBDRISC register. |
| | | 0 | A RESUME status has been detected. |
| | | 1 | An interrupt has not occurred. |



Copyright © 2024 Texas Instruments Incorporated

### 23.6.3.51 USB Device RESUME Raw Interrupt Mask Register (USBDRIM), offset 0x414

The USB device RESUME raw interrupt status register (USBDRIM) is the masked interrupt status register. On a read, this register gives the current masked status value of the corresponding interrupt. A write has no effect.

**Mode(s):** Host Device

USBDRIM is shown in Figure 23-63 and described in Table 23-67.

**Figure 23-63. USB Device RESUME Raw Interrupt Status Register (USBDRRIS)**

| 31 | 1 | 0 |
|---|---|---|
| Reserved | | RESUME |
| R-0 | | R-0 |

LEGEND: R = Read only; *-n* = value after reset

**Table 23-67. USB Device RESUME Raw Interrupt Status Register (USBDRRIS) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-1 | Reserved | 0 | Reserved. Reset is 0x0000.000. |
| 0 | PF | | RESUME Interrupt Mask |
| | | 0 | The raw interrupt signal from a detected RESUME is sent to the interrupt controller. This bit should only be set when a SUSPEND has been detected (the SUSPEND bit in the USBIS register is set). |
| | | 1 | A detected RESUME does not affect the interrupt status. |



Copyright © 2024 Texas Instruments Incorporated

#### 23.6.3.52 USB Device RESUME Interrupt Status and Clear Register (USBDRISC), offset 0x418

The USB device RESUME interrupt status and clear register (USBDRRIS) is the raw interrupt clear register. On a write of 1, the corresponding interrupt is cleared. A write of 0 has no effect.

**Mode(s):** Host Device

USBDRISC is shown in Figure 23-64 and described in Table 23-68.

**Figure 23-64. USB Device RESUME Interrupt Status and Clear Register (USBDRISC)**

| 31 | 1 | 0 |
|---|---|---|
| Reserved | | RESUME |
| R-0 | | R/W1C |

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

**Table 23-68. USB Device RESUME Interrupt Status and Clear Register (USBDRISC) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-1 | Reserved | 0 | Reserved. Reset is 0x0000.000. |
| 0 | RESUME | | RESUME Interrupt Status and Clear.<br>This bit is cleared by writing a 1. Clearing this bit also clears the RESUME bit in the USBDRCRIS register. |
| | | 0 | The RESUME bits in the USBDRRIS and USBDRCIM registers are set, providing an interrupt to the interrupt controller. |
| | | 1 | No interrupt has occurred or the interrupt is masked. |



Copyright © 2024 Texas Instruments Incorporated

#### 23.6.3.53 USB General-Purpose Control and Status Register (USBGPCS), offset 0x41C

The USB general-purpose control and status register (USBGPCS) provides the state of the internal ID signal.

When the USB controller is used as either a dedicated Host or Device, the DEVMODOTG and DEVMOD bits in the USB General-Purpose Control and Status (USBGPCS) register should be used to connect the ID inputs to fixed levels internally. For proper self-powered Device operation, the VBUS value must be monitored to assure that if the Host removes VBUS, the self-powered Device disables the D+/D- pull-up resistors. This function can be accomplished by connecting a standard GPIO to VBUS.

**Mode(s):** Host Device

USBGPCS is shown in Figure 23-65 and described in Table 23-69.

**Figure 23-65. USB General-Purpose Control and Status Register (USBGPCS)**

| 31 | 2 | 1 | 0 |
|---|---|---|---|
| Reserved | | DEVMODOTG | DEVMOD |
| R-0 | | R/W-0 | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

**Table 23-69. USB General-Purpose Control and Status Register (USBGPCS) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-2 | Reserved | 0 | Reserved. Reset is 0x0000.000. |
| 1 | DEVMODOTG | | Enable Device Mode. This bit enables the DEVMOD bit to control the state of the internal ID signal in OTG mode. |
| | | 0 | The mode is specified by the state of the internal ID signal. |
| | | 1 | This bit enables the DEVMOD bit to control the internal ID signal. |
| 0 | DEVMOD | | Device Mode This bit specifies the state of the internal ID signal in Host mode and in OTG mode when the DEVMODOTG bit is set.<br>In Device mode this bit is ignored (assumed set). |
| | | 0 | Host mode |
| | | 1 | Device mode |



Copyright © 2024 Texas Instruments Incorporated

### 23.6.3.54 USB DMA Select Register (USBDMASEL), offset 0x450

The USB DMA select 32-bit register (USBDMASEL) specifies whether the unmasked interrupt status of the ID value is valid.

**Mode(s):** Host Device

USBDMASEL is shown in Figure 23-66 and described in Table 23-70.

**Figure 23-66. USB DMA Select Register (USBDMASEL)**

| 31 | 24 | 23 | 20 | 19 | 16 |
|---|---|---|---|---|---|
| Reserved | | DMACTX | | DMACRX | |
| R/0 | | R/W-0 | | R/W-0 | |

| 15 | 12 | 11 | 8 | 7 | 4 | 3 | 0 |
|---|---|---|---|---|---|---|---|
| DMABTX | | DMABRX | | DMAATX | | DMAARX | |
| R/W-0 | | R/W-0 | | R/W-0 | | R/W-0 | |

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

**Table 23-70. USB DMA Select Register (USBDMASEL) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-24 | Reserved | 0 | Reserved. Reset is 0x0000.000. |
| 23-20 | DMACTX | | DMA C TX Select specifies the TX mapping of the third USB endpoint on DMA channel 5 (primary assignment). |
| | | 0h | Reserved |
| | | 1h | Endpoint 1 TX |
| | | 2h | Endpoint 2 TX |
| | | 3h | Endpoint 3 TX |
| 19-16 | DMACRX | | DMA C RX Select specifies the RX and TX mapping of the third USB endpoint on DMA channel 4 (primary assignment). |
| | | 0h | Reserved |
| | | 1h | Endpoint 1 RX |
| | | 2h | Endpoint 2 RX |
| | | 3h | Endpoint 3 RX |
| 15-12 | DMABTX | | DMA B TX Select specifies the TX mapping of the second USB endpoint on DMA channel 3 (primary assignment). |
| | | 0h | Reserved |
| | | 1h | Endpoint 1 TX |
| | | 2h | Endpoint 2 TX |
| | | 3h | Endpoint 3 TX |
| 11-8 | DMABRX | | DMA B RX Select Specifies the RX mapping of the second USB endpoint on DMA channel 2 (primary assignment). |
| | | 0h | Reserved |
| | | 1h | Endpoint 1 RX |
| | | 2h | Endpoint 2 RX |
| | | 3h | Endpoint 3 RX |



Copyright © 2024 Texas Instruments Incorporated

**Table 23-70. USB DMA Select Register (USBDMASEL) Field Descriptions (continued)**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7-4 | DMAATX | | DMA A TX Select specifies the TX mapping of the first USB endpoint on DMA channel 1 (primary assignment). |
| | | 0h | Reserved |
| | | 1h | Endpoint 1 TX |
| | | 2h | Endpoint 2 TX |
| | | 3h | Endpoint 3 TX |
| 3-0 | DMAARX | | DMA A RX Select specifies the RX mapping of the first USB endpoint on DMA channel 0 (primary assignment). |
| | | 0h | Reserved |
| | | 1h | Endpoint 1 RX |
| | | 2h | Endpoint 2 RX |
| | | 3h | Endpoint 3 RX |



Copyright © 2024 Texas Instruments Incorporated

### 23.6.4 USB Registers to Driverlib Functions

**Table 23-71. USB Registers to Driverlib Functions**

| File | Driverlib Function |
|---|---|
| **FADDR** | |
| usb.c | USBDevAddrSet |
| usb.c | USBDevAddrGet |
| **POWER** | |
| usb.c | USBHostSuspend |
| usb.c | USBHostReset |
| usb.c | USBHostResume |
| usb.c | USBDevConnect |
| usb.c | USBDevDisconnect |
| usb.c | USBPHYPowerOff |
| usb.c | USBPHYPowerOn |
| **TXIS** | |
| usb.c | USBIntStatus |
| usb.c | USBIntStatusEndpoint |
| **RXIS** | |
| usb.c | USBIntStatus |
| usb.c | USBIntStatusEndpoint |
| **TXIE** | |
| usb.c | USBIntDisableEndpoint |
| usb.c | USBIntEnableEndpoint |
| **RXIE** | |
| usb.c | USBIntDisableEndpoint |
| usb.c | USBIntEnableEndpoint |
| **IS** | |
| usb.c | USBIntStatus |
| usb.c | USBIntStatusControl |
| **IE** | |
| usb.c | USBIntDisableControl |
| usb.c | USBIntEnableControl |
| **FRAME** | |
| usb.c | USBFrameNumberGet |
| **EPIDX** | |
| usb.c | USBIndexWrite |
| usb.c | USBIndexRead |
| **TEST** | |
| - | |
| **FIFO0** | |
| usb.c | USBEndpointDataGet |
| usb.c | USBEndpointDataPut |
| usb.c | USBFIFOAddrGet |
| **FIFO1** | |
| - | |
| **FIFO2** | |



Copyright © 2024 Texas Instruments Incorporated

**Table 23-71. USB Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| - | |
| **FIFO3** | |
| - | |
| **FIFO4** | |
| - | |
| **FIFO5** | |
| - | |
| **FIFO6** | |
| - | |
| **FIFO7** | |
| - | |
| **FIFO8** | |
| - | |
| **FIFO9** | |
| - | |
| **FIFO10** | |
| - | |
| **FIFO11** | |
| - | |
| **FIFO12** | |
| - | |
| **FIFO13** | |
| - | |
| **FIFO14** | |
| - | |
| **FIFO15** | |
| - | |
| **DEVCTL** | |
| usb.c | USBHostSpeedGet |
| usb.c | USBOTGSessionRequest |
| usb.c | USBModeGet |
| **TXFIFOSZ** | |
| usb.c | USBFIFOConfigSet |
| usb.c | USBFIFOConfigGet |
| **RXFIFOSZ** | |
| usb.c | USBFIFOConfigSet |
| usb.c | USBFIFOConfigGet |
| **TXFIFOADD** | |
| usb.c | USBFIFOConfigSet |
| usb.c | USBFIFOConfigGet |
| **RXFIFOADD** | |
| usb.c | USBFIFOConfigSet |
| usb.c | USBFIFOConfigGet |
| **CONTIM** | |
| - | |



Copyright © 2024 Texas Instruments Incorporated

**Table 23-71. USB Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| **VPLEN** | |
| - | |
| **FSEOF** | |
| - | |
| **LSEOF** | |
| - | |
| **TXFUNCADDR0** | |
| usb.c | USBHostAddrSet |
| usb.c | USBHostAddrGet |
| **TXHUBADDR0** | |
| usb.c | USBHostHubAddrSet |
| usb.c | USBHostHubAddrGet |
| **TXHUBPORT0** | |
| - | |
| **TXFUNCADDR1** | |
| - | |
| **TXHUBADDR1** | |
| - | |
| **TXHUBPORT1** | |
| - | |
| **RXFUNCADDR1** | |
| - | |
| **RXHUBADDR1** | |
| - | |
| **RXHUBPORT1** | |
| - | |
| **TXFUNCADDR2** | |
| - | |
| **TXHUBADDR2** | |
| - | |
| **TXHUBPORT2** | |
| - | |
| **RXFUNCADDR2** | |
| - | |
| **RXHUBADDR2** | |
| - | |
| **RXHUBPORT2** | |
| - | |
| **TXFUNCADDR3** | |
| - | |
| **TXHUBADDR3** | |
| - | |
| **TXHUBPORT3** | |
| - | |
| **RXFUNCADDR3** | |



Copyright © 2024 Texas Instruments Incorporated

**Table 23-71. USB Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| - | |
| **RXHUBADDR3** | |
| - | |
| **RXHUBPORT3** | |
| - | |
| **TXFUNCADDR4** | |
| - | |
| **TXHUBADDR4** | |
| - | |
| **TXHUBPORT4** | |
| - | |
| **RXFUNCADDR4** | |
| - | |
| **RXHUBADDR4** | |
| - | |
| **RXHUBPORT4** | |
| - | |
| **TXFUNCADDR5** | |
| - | |
| **TXHUBADDR5** | |
| - | |
| **TXHUBPORT5** | |
| - | |
| **RXFUNCADDR5** | |
| - | |
| **RXHUBADDR5** | |
| - | |
| **RXHUBPORT5** | |
| - | |
| **TXFUNCADDR6** | |
| - | |
| **TXHUBADDR6** | |
| - | |
| **TXHUBPORT6** | |
| - | |
| **RXFUNCADDR6** | |
| - | |
| **RXHUBADDR6** | |
| - | |
| **RXHUBPORT6** | |
| - | |
| **TXFUNCADDR7** | |
| - | |
| **TXHUBADDR7** | |
| - | |



Copyright © 2024 Texas Instruments Incorporated

**Table 23-71. USB Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| **TXHUBPORT7** | |
| - | |
| **RXFUNCADDR7** | |
| - | |
| **RXHUBADDR7** | |
| - | |
| **RXHUBPORT7** | |
| - | |
| **TXFUNCADDR8** | |
| - | |
| **TXHUBADDR8** | |
| - | |
| **TXHUBPORT8** | |
| - | |
| **RXFUNCADDR8** | |
| - | |
| **RXHUBADDR8** | |
| - | |
| **RXHUBPORT8** | |
| - | |
| **TXFUNCADDR9** | |
| - | |
| **TXHUBADDR9** | |
| - | |
| **TXHUBPORT9** | |
| - | |
| **RXFUNCADDR9** | |
| - | |
| **RXHUBADDR9** | |
| - | |
| **RXHUBPORT9** | |
| - | |
| **TXFUNCADDR10** | |
| - | |
| **TXHUBADDR10** | |
| - | |
| **TXHUBPORT10** | |
| - | |
| **RXFUNCADDR10** | |
| - | |
| **RXHUBADDR10** | |
| - | |
| **RXHUBPORT10** | |
| - | |
| **TXFUNCADDR11** | |



Copyright © 2024 Texas Instruments Incorporated

**Table 23-71. USB Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| - | |
| **TXHUBADDR11** | |
| - | |
| **TXHUBPORT11** | |
| - | |
| **RXFUNCADDR11** | |
| - | |
| **RXHUBADDR11** | |
| - | |
| **RXHUBPORT11** | |
| - | |
| **TXFUNCADDR12** | |
| - | |
| **TXHUBADDR12** | |
| - | |
| **TXHUBPORT12** | |
| - | |
| **RXFUNCADDR12** | |
| - | |
| **RXHUBADDR12** | |
| - | |
| **RXHUBPORT12** | |
| - | |
| **TXFUNCADDR13** | |
| - | |
| **TXHUBADDR13** | |
| - | |
| **TXHUBPORT13** | |
| - | |
| **RXFUNCADDR13** | |
| - | |
| **RXHUBADDR13** | |
| - | |
| **RXHUBPORT13** | |
| - | |
| **TXFUNCADDR14** | |
| - | |
| **TXHUBADDR14** | |
| - | |
| **TXHUBPORT14** | |
| - | |
| **RXFUNCADDR14** | |
| - | |
| **RXHUBADDR14** | |
| - | |



Copyright © 2024 Texas Instruments Incorporated

**Table 23-71. USB Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| **RXHUBPORT14** | |
| - | |
| **TXFUNCADDR15** | |
| - | |
| **TXHUBADDR15** | |
| - | |
| **TXHUBPORT15** | |
| - | |
| **RXFUNCADDR15** | |
| - | |
| **RXHUBADDR15** | |
| - | |
| **RXHUBPORT15** | |
| - | |
| **CSRL0** | |
| usb.c | USBHostEndpointStatusClear |
| usb.c | USBDevEndpointStatusClear |
| usb.c | USBDevEndpointStall |
| usb.c | USBDevEndpointStallClear |
| usb.c | USBEndpointDataAvail |
| usb.c | USBEndpointDataGet |
| usb.c | USBDevEndpointDataAck |
| usb.c | USBHostEndpointDataAck |
| usb.c | USBEndpointDataPut |
| usb.c | USBEndpointDataSend |
| usb.c | USBFIFOFlush |
| usb.c | USBHostRequestIN |
| usb.c | USBHostRequestINClear |
| usb.c | USBHostRequestStatus |
| **CSRH0** | |
| usb.c | USBHostEndpointDataToggle |
| usb.c | USBFIFOFlush |
| **COUNT0** | |
| usb.c | USBEndpointDataAvail |
| usb.c | USBEndpointDataGet |
| **TYPE0** | |
| usb.c | USBHostEndpointConfig |
| usb.c | USBHostHubAddrSet |
| **NAKLMT** | |
| usb.c | USBHostEndpointConfig |
| **TXMAXP1** | |
| usb.c | USBHostEndpointConfig |
| usb.c | USBDevEndpointConfigSet |
| usb.c | USBDevEndpointConfigGet |
| **TXCSRL1** | |



Copyright © 2024 Texas Instruments Incorporated

**Table 23-71. USB Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| usb.c | USBEndpointStatus |
| usb.c | USBHostEndpointStatusClear |
| usb.c | USBDevEndpointStatusClear |
| usb.c | USBEndpointDataToggleClear |
| usb.c | USBDevEndpointStall |
| usb.c | USBDevEndpointStallClear |
| usb.c | USBDevEndpointConfigSet |
| usb.c | USBFIFOFlush |
| **TXCSRH1** | |
| usb.c | USBHostEndpointDataToggle |
| usb.c | USBHostEndpointConfig |
| usb.c | USBDevEndpointConfigSet |
| usb.c | USBDevEndpointConfigGet |
| usb.c | USBEndpointDMAConfigSet |
| usb.c | USBEndpointDMAEnable |
| usb.c | USBEndpointDMADisable |
| **RXMAXP1** | |
| usb.c | USBHostEndpointConfig |
| usb.c | USBDevEndpointConfigSet |
| usb.c | USBDevEndpointConfigGet |
| **RXCSRL1** | |
| usb.c | USBEndpointStatus |
| usb.c | USBHostEndpointStatusClear |
| usb.c | USBDevEndpointStatusClear |
| usb.c | USBEndpointDataToggleClear |
| usb.c | USBDevEndpointStall |
| usb.c | USBDevEndpointStallClear |
| usb.c | USBDevEndpointConfigSet |
| usb.c | USBEndpointDataAvail |
| usb.c | USBEndpointDataGet |
| usb.c | USBDevEndpointDataAck |
| usb.c | USBHostEndpointDataAck |
| usb.c | USBFIFOFlush |
| usb.c | USBHostRequestIN |
| usb.c | USBHostRequestINClear |
| **RXCSRH1** | |
| usb.c | USBHostEndpointDataToggle |
| usb.c | USBHostEndpointConfig |
| usb.c | USBDevEndpointConfigSet |
| usb.c | USBDevEndpointConfigGet |
| usb.c | USBEndpointDMAConfigSet |
| usb.c | USBEndpointDMAEnable |
| usb.c | USBEndpointDMADisable |
| **RXCOUNT1** | |
| - | |



Copyright © 2024 Texas Instruments Incorporated

**Table 23-71. USB Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| **TXTYPE1** | |
| usb.c | USBHostEndpointConfig |
| **TXINTERVAL1** | |
| usb.c | USBHostEndpointConfig |
| **RXTYPE1** | |
| usb.c | USBHostEndpointConfig |
| **RXINTERVAL1** | |
| usb.c | USBHostEndpointConfig |
| **TXMAXP2** | |
| - | |
| **TXCSRL2** | |
| - | |
| **TXCSRH2** | |
| - | |
| **RXMAXP2** | |
| - | |
| **RXCSRL2** | |
| - | |
| **RXCSRH2** | |
| - | |
| **RXCOUNT2** | |
| - | |
| **TXTYPE2** | |
| - | |
| **TXINTERVAL2** | |
| - | |
| **RXTYPE2** | |
| - | |
| **RXINTERVAL2** | |
| - | |
| **TXMAXP3** | |
| - | |
| **TXCSRL3** | |
| - | |
| **TXCSRH3** | |
| - | |
| **RXMAXP3** | |
| - | |
| **RXCSRL3** | |
| - | |
| **RXCSRH3** | |
| - | |
| **RXCOUNT3** | |
| - | |
| **TXTYPE3** | |



Copyright © 2024 Texas Instruments Incorporated

**Table 23-71. USB Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| - | |
| **TXINTERVAL3** | |
| - | |
| **RXTYPE3** | |
| - | |
| **RXINTERVAL3** | |
| - | |
| **TXMAXP4** | |
| - | |
| **TXCSRL4** | |
| - | |
| **TXCSRH4** | |
| - | |
| **RXMAXP4** | |
| - | |
| **RXCSRL4** | |
| - | |
| **RXCSRH4** | |
| - | |
| **RXCOUNT4** | |
| - | |
| **TXTYPE4** | |
| - | |
| **TXINTERVAL4** | |
| - | |
| **RXTYPE4** | |
| - | |
| **RXINTERVAL4** | |
| - | |
| **TXMAXP5** | |
| - | |
| **TXCSRL5** | |
| - | |
| **TXCSRH5** | |
| - | |
| **RXMAXP5** | |
| - | |
| **RXCSRL5** | |
| - | |
| **RXCSRH5** | |
| - | |
| **RXCOUNT5** | |
| - | |
| **TXTYPE5** | |
| - | |



Copyright © 2024 Texas Instruments Incorporated

**Table 23-71. USB Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| **TXINTERVAL5** | |
| - | |
| **RXTYPE5** | |
| - | |
| **RXINTERVAL5** | |
| - | |
| **TXMAXP6** | |
| - | |
| **TXCSRL6** | |
| - | |
| **TXCSRH6** | |
| - | |
| **RXMAXP6** | |
| - | |
| **RXCSRL6** | |
| - | |
| **RXCSRH6** | |
| - | |
| **RXCOUNT6** | |
| - | |
| **TXTYPE6** | |
| - | |
| **TXINTERVAL6** | |
| - | |
| **RXTYPE6** | |
| - | |
| **RXINTERVAL6** | |
| - | |
| **TXMAXP7** | |
| - | |
| **TXCSRL7** | |
| - | |
| **TXCSRH7** | |
| - | |
| **RXMAXP7** | |
| - | |
| **RXCSRL7** | |
| - | |
| **RXCSRH7** | |
| - | |
| **RXCOUNT7** | |
| - | |
| **TXTYPE7** | |
| - | |
| **TXINTERVAL7** | |



Copyright © 2024 Texas Instruments Incorporated

**Table 23-71. USB Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| - | |
| **RXTYPE7** | |
| - | |
| **RXINTERVAL7** | |
| - | |
| **TXMAXP8** | |
| - | |
| **TXCSRL8** | |
| - | |
| **TXCSRH8** | |
| - | |
| **RXMAXP8** | |
| - | |
| **RXCSRL8** | |
| - | |
| **RXCSRH8** | |
| - | |
| **RXCOUNT8** | |
| - | |
| **TXTYPE8** | |
| - | |
| **TXINTERVAL8** | |
| - | |
| **RXTYPE8** | |
| - | |
| **RXINTERVAL8** | |
| - | |
| **TXMAXP9** | |
| - | |
| **TXCSRL9** | |
| - | |
| **TXCSRH9** | |
| - | |
| **RXMAXP9** | |
| - | |
| **RXCSRL9** | |
| - | |
| **RXCSRH9** | |
| - | |
| **RXCOUNT9** | |
| - | |
| **TXTYPE9** | |
| - | |
| **TXINTERVAL9** | |
| - | |



Copyright © 2024 Texas Instruments Incorporated

**Table 23-71. USB Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| **RXTYPE9** | |
| - | |
| **RXINTERVAL9** | |
| - | |
| **TXMAXP10** | |
| - | |
| **TXCSRL10** | |
| - | |
| **TXCSRH10** | |
| - | |
| **RXMAXP10** | |
| - | |
| **RXCSRL10** | |
| - | |
| **RXCSRH10** | |
| - | |
| **RXCOUNT10** | |
| - | |
| **TXTYPE10** | |
| - | |
| **TXINTERVAL10** | |
| - | |
| **RXTYPE10** | |
| - | |
| **RXINTERVAL10** | |
| - | |
| **TXMAXP11** | |
| - | |
| **TXCSRL11** | |
| - | |
| **TXCSRH11** | |
| - | |
| **RXMAXP11** | |
| - | |
| **RXCSRL11** | |
| - | |
| **RXCSRH11** | |
| - | |
| **RXCOUNT11** | |
| - | |
| **TXTYPE11** | |
| - | |
| **TXINTERVAL11** | |
| - | |
| **RXTYPE11** | |



Copyright © 2024 Texas Instruments Incorporated

**Table 23-71. USB Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| - | |
| **RXINTERVAL11** | |
| - | |
| **TXMAXP12** | |
| - | |
| **TXCSRL12** | |
| - | |
| **TXCSRH12** | |
| - | |
| **RXMAXP12** | |
| - | |
| **RXCSRL12** | |
| - | |
| **RXCSRH12** | |
| - | |
| **RXCOUNT12** | |
| - | |
| **TXTYPE12** | |
| - | |
| **TXINTERVAL12** | |
| - | |
| **RXTYPE12** | |
| - | |
| **RXINTERVAL12** | |
| - | |
| **TXMAXP13** | |
| - | |
| **TXCSRL13** | |
| - | |
| **TXCSRH13** | |
| - | |
| **RXMAXP13** | |
| - | |
| **RXCSRL13** | |
| - | |
| **RXCSRH13** | |
| - | |
| **RXCOUNT13** | |
| - | |
| **TXTYPE13** | |
| - | |
| **TXINTERVAL13** | |
| - | |
| **RXTYPE13** | |
| - | |



Copyright © 2024 Texas Instruments Incorporated

**Table 23-71. USB Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| **RXINTERVAL13** | |
| - | |
| **TXMAXP14** | |
| - | |
| **TXCSRL14** | |
| - | |
| **TXCSRH14** | |
| - | |
| **RXMAXP14** | |
| - | |
| **RXCSRL14** | |
| - | |
| **RXCSRH14** | |
| - | |
| **RXCOUNT14** | |
| - | |
| **TXTYPE14** | |
| - | |
| **TXINTERVAL14** | |
| - | |
| **RXTYPE14** | |
| - | |
| **RXINTERVAL14** | |
| - | |
| **TXMAXP15** | |
| - | |
| **TXCSRL15** | |
| - | |
| **TXCSRH15** | |
| - | |
| **RXMAXP15** | |
| - | |
| **RXCSRL15** | |
| - | |
| **RXCSRH15** | |
| - | |
| **RXCOUNT15** | |
| - | |
| **TXTYPE15** | |
| - | |
| **TXINTERVAL15** | |
| - | |
| **RXTYPE15** | |
| - | |
| **RXINTERVAL15** | |



Copyright © 2024 Texas Instruments Incorporated

**Table 23-71. USB Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| - | |
| **RQPKTCOUNT1** | |
| usb.c | USBEndpointPacketCountSet |
| **RQPKTCOUNT2** | |
| - | |
| **RQPKTCOUNT3** | |
| - | |
| **RQPKTCOUNT4** | |
| - | |
| **RQPKTCOUNT5** | |
| - | |
| **RQPKTCOUNT6** | |
| - | |
| **RQPKTCOUNT7** | |
| - | |
| **RQPKTCOUNT8** | |
| - | |
| **RQPKTCOUNT9** | |
| - | |
| **RQPKTCOUNT10** | |
| - | |
| **RQPKTCOUNT11** | |
| - | |
| **RQPKTCOUNT12** | |
| - | |
| **RQPKTCOUNT13** | |
| - | |
| **RQPKTCOUNT14** | |
| - | |
| **RQPKTCOUNT15** | |
| - | |
| **RXDPKTBUFDIS** | |
| - | |
| **TXDPKTBUFDIS** | |
| - | |
| **EPC** | |
| usb.c | USBIntDisableControl |
| usb.c | USBIntEnableControl |
| usb.c | USBIntStatus |
| usb.c | USBIntStatusControl |
| usb.c | USBHostPwrConfig |
| usb.c | USBHostPwrFaultEnable |
| usb.c | USBHostPwrFaultDisable |
| usb.c | USBHostPwrEnable |
| usb.c | USBHostPwrDisable |



Copyright © 2024 Texas Instruments Incorporated

**Table 23-71. USB Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| **EPCRIS** | |
| - | |
| **EPCIM** | |
| usb.c | USBIntDisableControl |
| usb.c | USBIntEnableControl |
| **EPCISC** | |
| usb.c | USBIntStatus |
| usb.c | USBIntStatusControl |
| **DRRIS** | |
| - | |
| **DRIM** | |
| - | |
| **DRISC** | |
| - | |
| **GPCS** | |
| usb.c | USBHostMode |
| usb.c | USBDevMode |
| usb.c | USBOTGMode |
| **VDC** | |
| usb.c | USBHostPwrConfig |
| **VDCRIS** | |
| - | |
| **VDCIM** | |
| - | |
| **VDCISC** | |
| - | |
| **IDVRIS** | |
| - | |
| **IDVIM** | |
| usb.c | USBIntDisableControl |
| usb.c | USBIntEnableControl |
| **IDVISC** | |
| usb.c | USBIntStatus |
| usb.c | USBIntStatusControl |
| **DMASEL** | |
| usb.c | USBEndpointDMAChannel |
| **PP** | |
| - | |



Copyright © 2024 Texas Instruments Incorporated

Chapter 24

# Universal Parallel Port (uPP)

This chapter describes the features and functions of the Universal Parallel Port module (uPP).


**24.1 Introduction** ..... 2749
**24.2 Configuring Device Pins** ..... 2750
**24.3 Functional Description** ..... 2750
**24.4 IO Interface and System Requirements** ..... 2753
**24.5 UPP Registers** ..... 2763





Copyright © 2024 Texas Instruments Incorporated

## 24.1 Introduction

The universal parallel port (uPP) peripheral is a high-speed parallel interface with dedicated data lines and minimal control signals. It is designed to interface cleanly with high-speed analog-to-digital converters (ADCs) or digital-to-analog converters (DACs) with 8-bit data width. It can also be interconnected with field-programmable gate arrays (FPGAs) or other uPP devices to achieve high-speed digital data transfer. It can operate in receive mode or transmit mode (simplex mode).

This peripheral includes an internal DMA controller to maximize throughput and minimize CPU overhead during high-speed data transmission. All uPP transactions uses internal DMA to feed data to or retrieve data from the I/O channels. Even though there is only one I/O channel, DMA controller includes two DMA channels to supports data interleave mode, in which all DMA resources service a single I/O channel.

On this device, uPP is dedicated resource for CPU1 subsystem and CPU1, CPU1.CLA1 and CPU1.DMA have access to this module. There are two dedicated DATA RAMs (also known as MSG RAMs), each of 512B, tightly coupled with uPP module (one for each, TX and RX). These DATA RAMs are used to store bulk of data to avoid frequent interruption to CPU. Only CPU1 and CPU1.CLA1 has access to these DATA RAMs. Figure 24-1 shows the integration of uPP on this device.

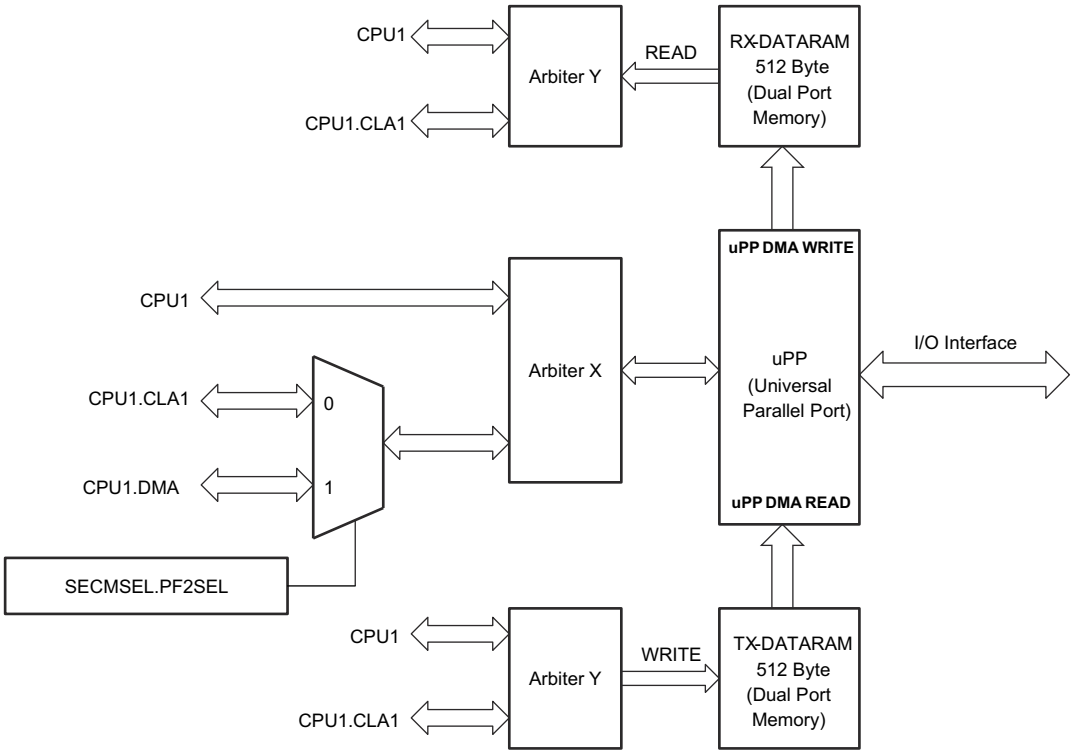



**Figure 24-1. uPP Integration**

**Note**

In some other TI devices, uPP IP is also called the Radio Peripheral Interface (RPI) module.

### *24.1.1 Features Supported*

- Supports mainstream high-speed data converters with parallel conversion interface.
- Supports mainstream high-speed streaming interface with frame START indication.
- Supports mainstream high-speed streaming interface with data ENABLE indication.
- Supports mainstream high-speed streaming interface with synchronization WAIT signal.
- Supports SDR (single-data-rate) or DDR (double-data-rate, interleaved) interface.
- Supports multiplexing of interleaved data in SDR transmit case.



Copyright © 2024 Texas Instruments Incorporated

- Supports de-multiplexing and multiplexing of interleaved data in DDR case.
- Supports I/O interface clock frequency up to 50 MHz for SDR case, and 25 MHz for DDR.
- Supports single-channel 8-bit input receive or output transmit mode.
- Max throughput is 50MB/s for pure read or pure write.
- Available as a DSP to FPGA general purpose streaming interface.

## 24.2 Configuring Device Pins

The GPIO mux registers must be configured to connect this peripheral to the device pins. To avoid glitches on the pins, the GPyGMUX bits must be configured first (while keeping the corresponding GPyMUX bits at the default of zero), followed by writing the GPyMUX register to the desired value.

Some IO functionality is defined by GPIO register settings independent of this peripheral. For input signals, the GPIO input qualification m,ust 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.

## 24.3 Functional Description

### *24.3.1 Functional Block Diagram*

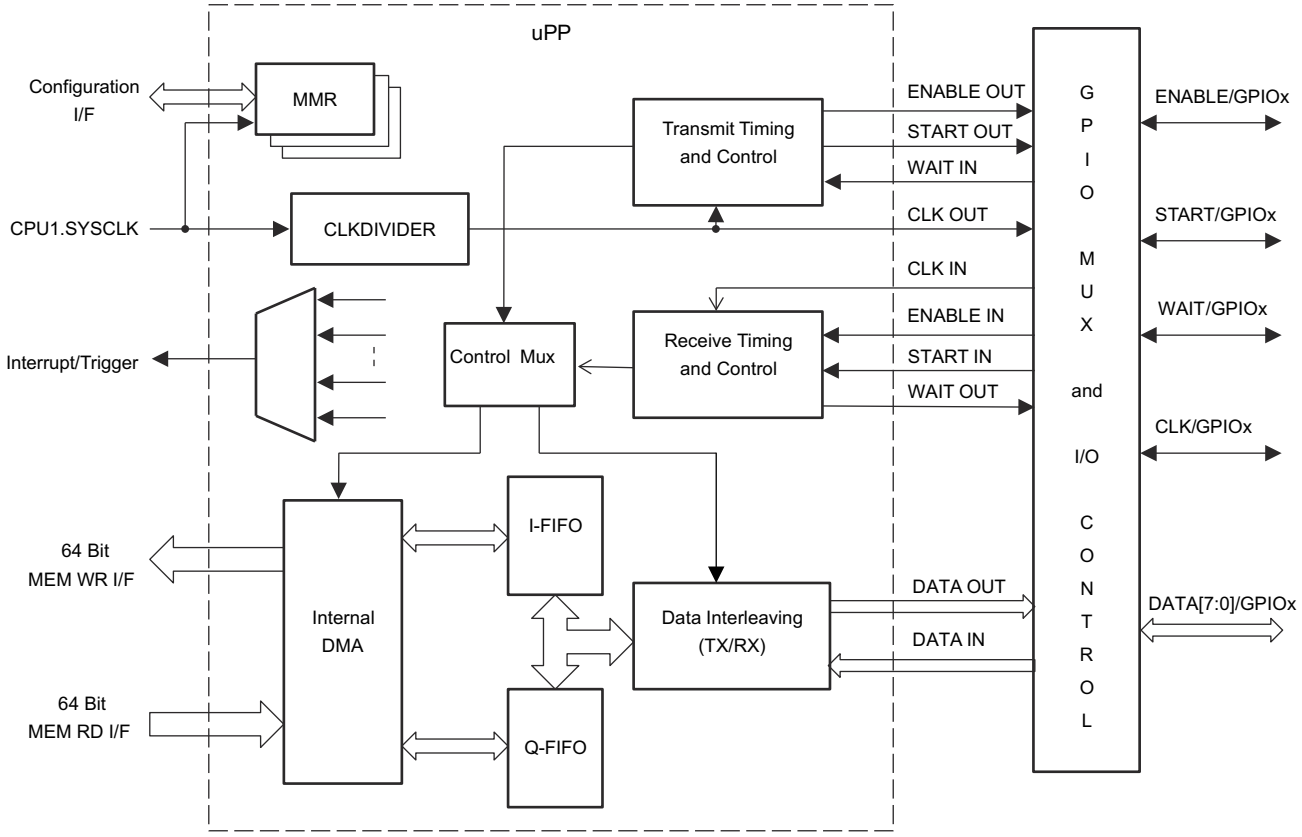



**Figure 24-2. Functional Block Diagram**

The uPP IP supports internal FIFO to store the data to and from IOs. To alleviate the load of data transfer from system memory to and from the uPP FIFOs, the IP supports internal DMA. The internal DMA has two channels: channel I and channel Q.



Copyright © 2024 Texas Instruments Incorporated

### 24.3.2 Data Flow

The following naming conventions are followed regarding channel numbering:

- Only one I/O channel and that is labeled as “channel A”
- DMA channel are labeled as “channel I and channel Q”.

Figure 24-3 explains the data flow for receive in SDR mode and DDR non-demux mode. In this case, only one DMA channel (channel I) get used.

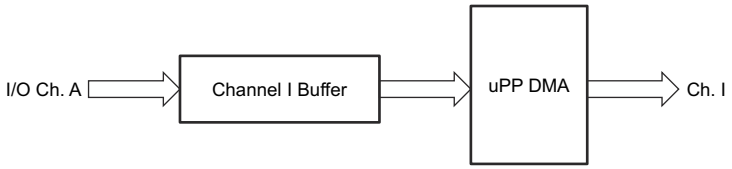



**Figure 24-3. RX in SDR or DDR (Non-Demux) Mode**

Figure 24-4 explains the data flow for receive in DDR demux mode. In this case, both DMA channels (channel I and channel Q) are used.

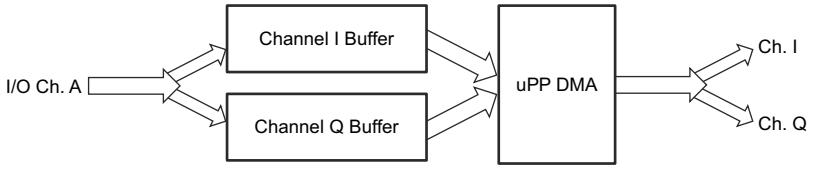



**Figure 24-4. RX in DDR (Demux) Mode**

Figure 24-5 explains the data flow for transmit in SDR non-interleave or DDR non-demux mode. In this case, only one DMA channel (channel I) get used.

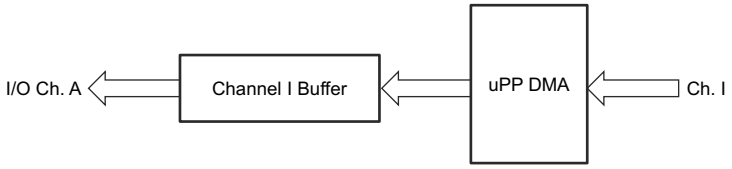



**Figure 24-5. TX in SDR (Non-Interleave) or DDR (Non-Demux) Mode**

Figure 24-6 explains the data flow for transmit in SDR interleave or DDR demux mode. In this case, both DMA channels (channel I and channel Q) are used.

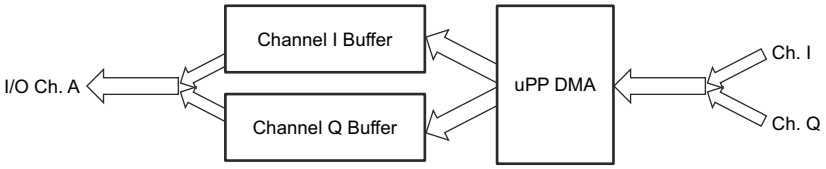



**Figure 24-6. TX in SDR (Interleave) or DDR (Demux) Mode**



Copyright © 2024 Texas Instruments Incorporated

### 24.3.3 Clock Generation and Control

The uPP peripheral uses three different clocks:

- A module clock that controls the internal logic and CPU interface. This is driven by CPU1.SYSCLK.
- A transmit clock (internally divided clock from the module clock) that runs the interface channel and drives the CLK pin out in transmit mode.
- A receive clock (input using CLK pin) that runs the interface channel in receive mode.

Figure 24-7 shows the output clock generation system in transmit mode.

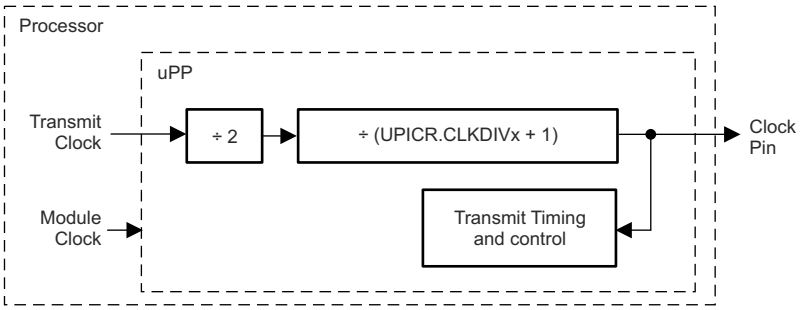



**Figure 24-7. IO Output Clock Generation for TX Mode**

Output Clock = Module Clock/(2 × (CLKDIVn + 1))

The fixed divisor restricts the maximum speed of the I/O clock to 1/2 the device CPU clock speed. For DDR mode operation, the CLK output frq must be one eighth (1/8) or less of module clock.

For receive mode, the channel requires an external clock to drive the CLK pin. The incoming clock is not divided, and the maximum allowed speed is one fourth (¼) the module clock speed for SDR mode and one eighth (1/8) the module clock speed for DDR mode. Figure 24-8 shows the clock generation system for a channel configured in receive mode.

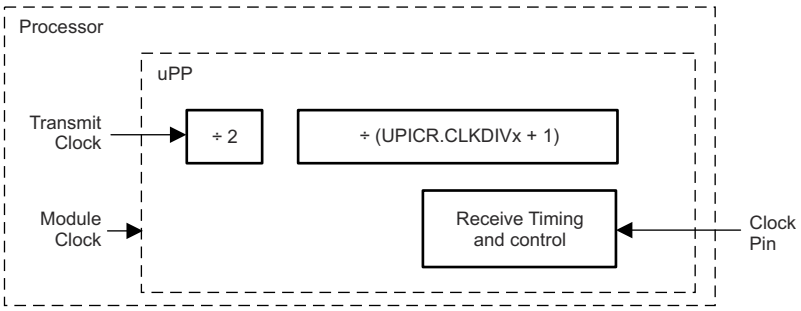



**Figure 24-8. IO Input Clock for RX Mode**



Copyright © 2024 Texas Instruments Incorporated

## 24.4 IO Interface and System Requirements

The uPP module provides interfacing logic to external streaming data, both inbound and outbound flow of data are supported. The interface protocol is a simple streaming interface and is built on top of the existing high-speed data converters, and captures most of the high-speed data converters protocol.

The input receive mode and output transmit mode protocol are symmetrical, and source synchronous, that is, clock is supplied by the transmitter and the same clock is used to receive the data.

Table 24-1 describes the functionality of all the input/output (IO) signals of uPP module.

**Table 24-1. uPP Signal Description**

| Signal | Description |
|---|---|
| CLK | • Transmit or receive clock<br>• Input in receive mode<br>• Output in transmit mode (Clock divider programmable) |
| START | • Input in receive mode<br>• Output in transmit mode<br>• Polarity programmable<br>• Indicates first data word of each line (frame)<br>• This is optional signal and if this feature is not used, external receiver can ignore this signal. |
| ENABLE | • Input when in receive mode<br>• Output in transmit mode<br>• Polarity programmable<br>• Indicates data received is valid, or data transmitted is valid<br>• Can be programmed to be ignored in receive mode but in transmit mode always drive active in sync with valid data. |
| WAIT | • Input in transmit mode<br>• Output in receive mode but always drive inactive by uPP in this case<br>• Polarity programmable<br>• In transmit mode when active, indicates that target is not ready to receive data. uPP stops transmitting data at next clock cycle right after the Wait signal is sampled as active. uPP module can be programmed to ignore this in transmit mode.<br>**Note**<br>The WAIT needs to be asserted for one full cycle of IO clock by external device. |
| DATA[7:0] | • 8-bit inputs in receive mode<br>• 8-bit output in transmit mode |

### *24.4.1 Pin Multiplexing*

Extensive pin multiplexing is used to accommodate the largest number of peripheral functions in the smallest possible package. Pin multiplexing is controlled using a combination of hardware configuration at device reset and software programmable register settings. To determine how pin multiplexing affects the uPP peripheral, see the device-specific data manual.

### *24.4.2 Internal DMA Controller Description*

The uPP peripheral includes an internal DMA controller separate from any device-level DMA. The internal DMA controller consists of two DMA channels, channel I and channel Q, which moves data to and from the uPP peripheral interface (I/O) channels in all operating modes. This section describes how to program the internal DMA channels.



Copyright © 2024 Texas Instruments Incorporated

#### 24.4.2.1 DMA Programming Concepts

The uPP internal DMA controller uses a simplified programming model similar to 2D transfers performed by any other DMA at system level. Each DMA channel may be configured with four parameters: window address, byte count, line count, and line offset address.

- Window Address (CHxDESC0.ADDR) - The location in uPP data memory of the first byte in the data buffer. When the uPP operates in receive mode, the DMA channel begins writing to this address as it takes incoming data from the uPP I/O channel. When the uPP operates in transmit mode, the DMA channel begins reading from this address and pass the data to the uPP I/O channel. The window address must be aligned to a 64-bit boundary (that is, the three LSBs must equal 0). Non-aligned addresses are automatically adjusted to a properly aligned value when written to configuration register.
- Byte Count (CHxDESC1.BCNT) - The number of bytes per line. The byte count must be an even number.
- Line Count (UPxDESC1.LCNT) - The number of lines per window. The total number of bytes transferred equals B x L, where B is the byte count per line and L is the line count.
- Line Offset Address (CHxDESC2.LOFFSET) - The offset address between the first byte in successive Lines. The line offset address cannot exceed 65528 (FFF8h) bytes, and must be aligned to a 64-bit boundary in memory (that is, the three LSBs must equal 0).

Figure 24-9 shows a typical DMA window defined by these parameters.

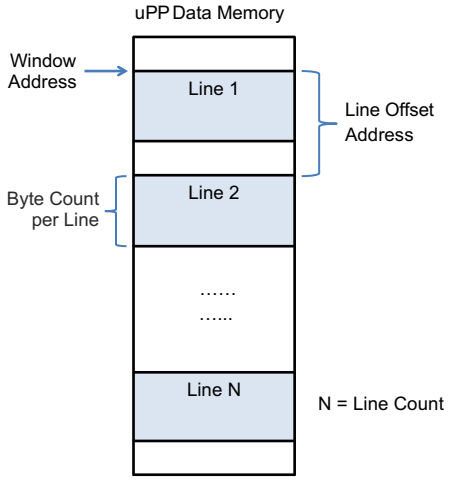



**Figure 24-9. Structure of DMA Window and Lines in Memory**

Certain values of the line offset address have special implications on the structure of the data buffer:

- Line Offset Address = Byte Count : Data buffer is a contiguous block in memory with size equal to (Line Count) x (Byte Count).
- Line Offset Address = 0 : Data buffer consists of a single line, with total size equal to Byte Count. If the I/O channel is configured in transmit mode, this line is transmitted (Line Count) consecutive times before the DMA transfer completes. If the I/O channel is configured in receive mode, the buffer is repeatedly written and overwritten by incoming data.

To program a DMA transfer, write the appropriate fields in the DMA channel descriptor registers for DMA Channel I or for DMA Channel Q. If the associated I/O channel is initialized and idle, the DMA transfer and I/O transaction begins immediately. Section 24.4.10 describes a step-by-step process for configuring the I/O channel and DMA channels to start a uPP transfer.

Each DMA channel allows a second descriptor to be queued while the previously programmed DMA transfer is still running. The PEND status bit in channel status register reports whether a new set of DMA parameters may be written to the DMA descriptor registers. Each DMA channel can have at most one active transfer and one queued transfer. This allows each I/O channel to perform uninterrupted, consecutive transactions across DMA transfer boundaries.



Copyright © 2024 Texas Instruments Incorporated

This DMA controller does not support automatically reloading DMA transfer descriptors. Each successive descriptor set must be explicitly written to the configuration registers by software. All uPP interrupt events originate in the internal DMA controller. Section 24.4.6 lists and explains all uPP interrupt events.

The internal DMA controller always writes data in bursts of 64 bytes. However, DMA read operations have configurable burst size, which may be set per channel using the RDSIZEI and RDSIZEQ bits in the uPP threshold configuration register. A DMA channel waits until the specified number of bytes leaves its internal buffer before performing another burst read from memory.

Note that the TXSIZEA bit in threshold configuration register is not DMA parameters; instead, it control transmit thresholds for the uPP interface channel.

#### 24.4.2.2 Data Interleave Mode

The data interleave mode is a special configuration that maps both DMA channels to a single interface I/O channel. There are two variants on data interleave mode, each with special conditions:

- Single Data Rate (SDR) Interleave
  - Single Data Rate (DRA = 0)
  - Transmit Only (MODE = 1)
  - SDR Transmit interleave enabled (SDRTXILA = 1)
- Double Data Rate (DDR) Interleave
  - Double Data Rate (DRA = 1)
  - Transmit or Receive (MODE = 0/1)
  - SDR Transmit interleave disabled (SDRTXILA = 0)
  - DDR interleave enabled (DEMUXA = 1)

In data interleave mode, I/O channel is associated with two data buffers, each serviced by its own DMA channel (I and Q). In SDR interleave mode, the START signal is used as a buffer selection line: START = 1 indicates that the current word comes from DMA Channel I; START = 0 indicates that the current word comes from DMA Channel Q. In DDR Interleave mode, the data buffers alternate every word beginning with Channel I: Channel I Word 0, Channel Q Word 0, Channel I Word 1, Channel Q Word 1, and so forth. Section 24.4.10 shows signal diagrams for both data interleave modes.

### *24.4.3 Protocol Description*

The uPP peripheral on this device has one I/O channel with 8 data lines, which can be configured to run in transmit or receive mode. A channel may also be configured to ignore certain control signals using the uPP interface configuration register. By default it uses all four control signals, unless otherwise configured.

#### 24.4.3.1 DATA[7:0] Signals

DATA[7:0] comprise the channel's entire data bus. In transmit mode, these pins are outputs that transmit data supplied by the channel's associated DMA channel. While the channel is idle, their behavior depends on the TRISENA bit in IFCFG register. These pins can be configured to drive an idle value (TRISENA = 0, VALA field in the uPP interface idle value register (IFIVAL)) or be in a high-impedance state while idle (TRISENA = 1). In receive mode, these pins are inputs that provide data to the channel's associated DMA channel.

#### 24.4.3.2 START Signal

The uPP transmitter asserts the START signal when it transfers the first word of a data line. A line is defined in terms of the channel's associated DMA channel; for more on DMA programming concepts, see Section 24.4.2. The START signal is active-high by default, but its polarity is controlled by the STARTPOLA bit in IFCFG register.

In transmit mode, START is an output signal and is always driven, in receive mode, START is an input signal and may be disabled using the STARTA bit in IFCFG register. When the channel is configured in transmit mode with data interleave enabled (SDRTXILA = 1 in CHCTL register), the START signal function changes completely. The START signal now asserts on every data word that is provided by DMA Channel I. For this alternative behavior, see Section 24.4.3.6.



Copyright © 2024 Texas Instruments Incorporated

#### 24.4.3.3 ENABLE

The uPP transmitter asserts the ENABLE signal when it transfers a valid data word. The ENABLE signal is active-high by default, but its polarity is controlled by the ENAPOLA bit in IFCFG register. In transmit mode, ENABLE is an output signal and is always driven; in receive mode, ENABLE is an input signal and may be disabled using the ENAA bit in IFCFG register.

#### 24.4.3.4 WAIT Signal

The WAIT signal allows the receiver to request a temporary halt in data transmission. When the receiver asserts WAIT, the transmitter responds by stopping transmission (starting with the next word) until WAIT is released. The receiver ignores all incoming data until WAIT is released. Once WAIT is released, the transmitter can resume transmission on the next word. Section 24.4.3 shows WAIT signal timing. The WAIT signal is active-high by default, but its polarity is controlled by the WAITPOLA bit in IFCFG register. In transmit mode, WAIT is an input signal and may be disabled using the WAITA bit in IFCFG register; in receive mode, WAIT is an output signal and always driven inactive by uPP.

#### 24.4.3.5 CLOCK Signal

The uPP transmitter drives the CLOCK signal to align all other uPP signals. By default, other signals align on the rising edge of CLOCK, but the polarity is controlled by the CLKINVA bit in IFCFG register. The active edges of CLOCK must always slightly precede transitions of other uPP signals. In transmit mode, CLOCK is an output signal; in receive mode, CLOCK is an input signal. For more information on clock generation and allowed frequencies, see Section 24.4.2.

#### 24.4.3.6 Signal Timing Diagrams

In the following diagrams, signals are marked (i) to indicate that the signals are inputs to the uPP peripheral and are marked (o) to indicate that the signals are outputs from the uPP peripheral. Data words from a single DMA channel are designated Dx, while data words that must come from a specific DMA channel are designated Ix or Qx to indicate DMA Channel I or Q, respectively. For more information on DMA channels and data interleave mode, see Section 24.4.2.

All signal diagrams are drawn with signal polarities in their default states. All signals except DATA are independently configurable in the uPP interface configuration register (IFCFG).

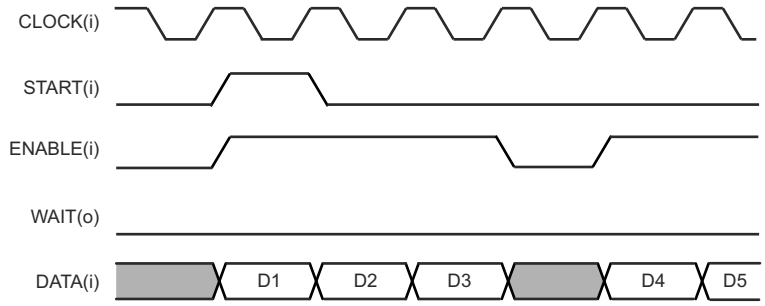



**Figure 24-10. uPP Receive in SDR Mode**

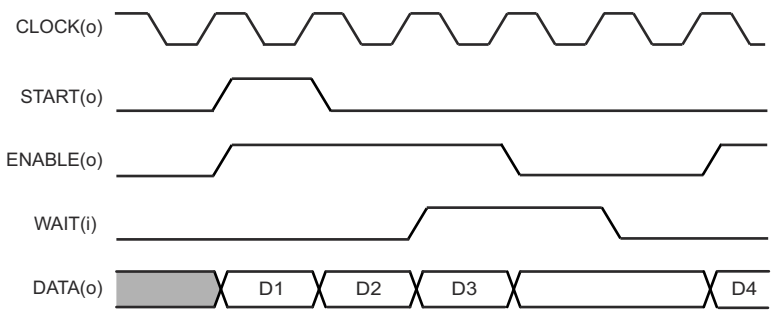



**Figure 24-11. uPP Transmit in SDR Mode**



Copyright © 2024 Texas Instruments Incorporated

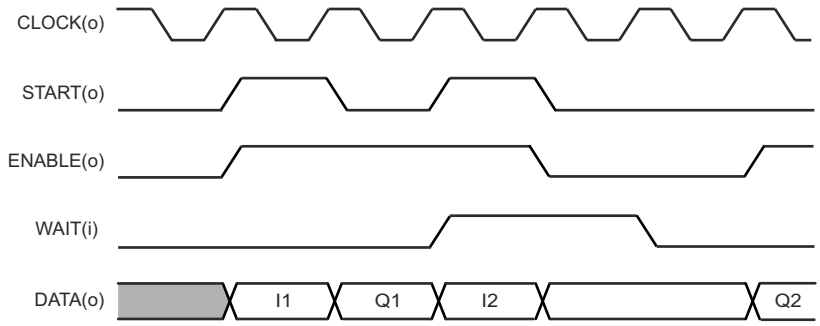



**Figure 24-12. uPP Transmit in SDR Mode – Interleaving**

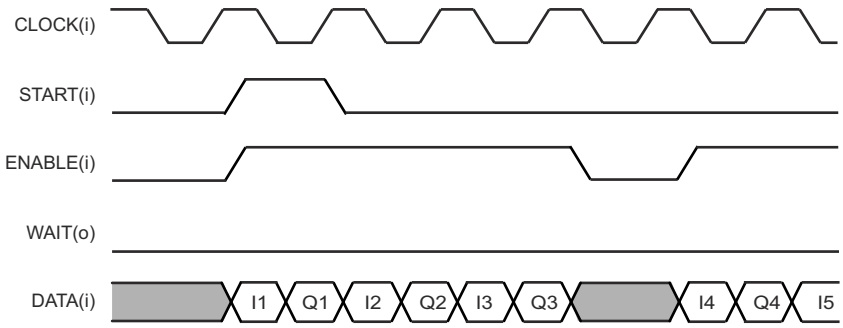



**Figure 24-13. uPP Receive DDR Case**

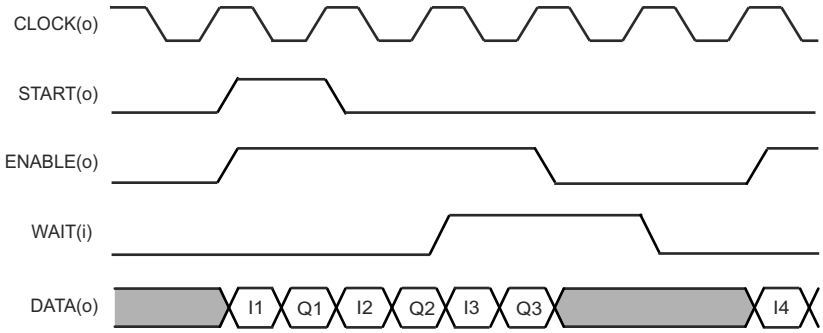



**Figure 24-14. uPP Transmit DDR Case**

**Note**

START asserts on every data word from DMA Channel I in interleaving mode.



Copyright © 2024 Texas Instruments Incorporated

### 24.4.4 Data Format

On this device, uPP only supports an 8-bit interface; hence, no data packing is performed. In this case, make the appropriate connection externally, if needed. For example, if a 4-bit ADC is used, then tie the upper 4-bits of the uPP data inputs to 0 or 1, as needed. If external FPGA is connected, then make sure that 8-bit data (or appropriate tie-offs) is driven out from the FPGA. By default, uPP supports little-endian data format but to support the data format of one of the TI devices (CC1260), a configuration bit is provided to select 16-bit big-endian mode also.

Figure 24-15 and Figure 24-16 describe how data goes out and comes in based on these modes.

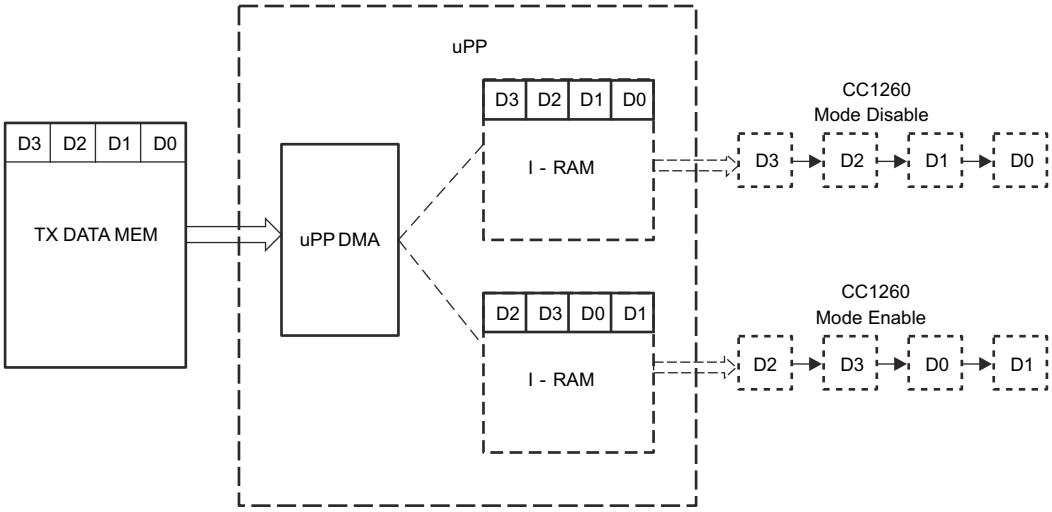



**Figure 24-15. uPP Tx Data Pattern in Non-Interleaved Mode**

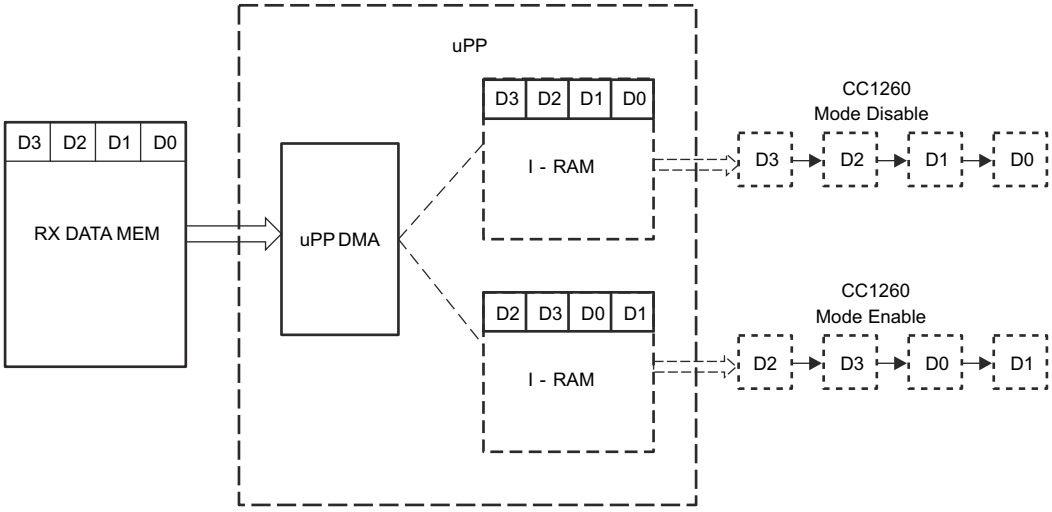



**Figure 24-16. uPP Rx Data Pattern in Non-Interleaved Mode**



Copyright © 2024 Texas Instruments Incorporated

### 24.4.5 Reset Considerations

#### 24.4.5.1 Software Reset

Software reset clears the uPP internal state machines but does not reset the contents of the uPP registers. The following procedure performs a software reset on the uPP peripheral:

1. Write the PEREN bit in the uPP PERCTL register to 0 (disables the uPP).
2. Poll the DMAST bit in uPP PERCTL register for activity; wait until DMA controller is inactive and idle.
3. Write the SOFTRST bit in uPP PERCTL register to 1 (places uPP in software reset).
4. Write the SOFTRST bit in uPP PERCTL register to 0 to (brings uPP out of software reset).

**Note**

In receive mode, uPP IO input clock must be running to have effect of soft reset.

#### 24.4.5.2 Hardware Reset

When the processor XRSn (or CPU1.SYSRSn) pin is asserted, the entire processor is reset and is held in the reset state until the RESET pin is released. As part of a device reset, the uPP state machines are reset, and the uPP registers are forced to their default states. For default states, see the register section.

**Note**

There is no SW method to apply a hard reset to the UPP module on this device.

### 24.4.6 Interrupt Support

The uPP peripheral generates multiple interrupt events, all tied to internal DMA Channels I and Q. The uPP peripheral automatically combines all interrupt events into a single chip-level interrupt driven to CPU1 and CPU1.CLA1. Individual events may be enabled using the uPP interrupt enable set register (INTENSET) and disabled using the uPP interrupt enable clear register (INTENCLR). Only enabled events generate interrupts and assert bits in the enabled interrupt status register (ENINTST). Disabled events do not generate interrupts but do assert bits in the raw interrupt status register (RAWINTST). An interrupt service routine (ISR) may be assigned to handle uPP CPU-level interrupts using the interrupt controller module. If uPP events occur in close proximity to one another, a single CPU interrupt (and a single call to the ISR) may represent multiple interrupt events. Thus, the uPP ISR must meet certain structural requirements:

- The ISR must be able to handle multiple events before returning
- The ISR must handle any subsequent events that occur after it is called but before it returns

Like CPU ISR, CLA tasks can be assigned based on these interrupt events.

#### 24.4.6.1 End of Line (EOL) Event

This event occurs each time when DMA channel reaches the end of a line in the data window. Note that if the associated uPP interface channel is operating in transmit mode, this event may occur before the line's final bytes are actually transmitted over the data pins.

For small line size and fast data transfer, it is possible to "miss" EOL events if they occur faster than the user's code can handle them. This does not hinder uPP operation; the uPP peripheral continue processing data uninterrupted until the EOW event or some error condition is encountered.



Copyright © 2024 Texas Instruments Incorporated

#### 24.4.6.2 End of Window (EOW) Event

This event occurs when the DMA channel reaches the end of the current data window. Note that if the associated uPP interface channel is operating in transmit mode, this event can occur shortly before the window's final bytes are actually transmitted over the data pins.

When an EOW event occurs, the DMA channel automatically begins the next DMA transfer if one has been pre-programmed into the channel descriptor registers. If no new transfer is preprogrammed, the DMA channel becomes idle. For small window size and fast data transfer, code overhead can make it impossible to maintain a constant flow of data through the uPP interface channel. This problem can be solved by increasing the DMA window size or decreasing the peripheral clock speed.

#### 24.4.6.3 Underrun or Overflow (UOR) Event

This event occurs when the DMA channel fails to keep up with incoming or outgoing data on the associated interface channel. Typically, this error indicates that background system activity has interfered with normal operation of the peripheral. This error does not occur simply when a channel is allowed to idle. After encountering this error, the uPP peripheral must be reset when this event occurs.

This error must primarily occur when operating the uPP at high speed with significant system loading. To avoid this error, run the uPP at slower speeds or reduce background activity, such as non-uPP peripheral or DMA transactions. Additional tuning tips are given in Section 24.4.10.1.

#### 24.4.6.4 DMA Programming Error (DPE) Event

This event occurs when the DMA channel descriptors are programmed while the PEND bit in the uPP DMA channel status register is set to 1. A channel's descriptors must only be programmed while the channel's PEND bit is cleared to 0.

### *24.4.7 Emulation Considerations*

The uPP peripheral stops running if any of three conditions are met:

- Peripheral Disable - EN bit in the uPP peripheral control register (PERCTL) is 0.
- Clock Stop – Clock to uPP is disabled using PCLKCR.
- Emulation Suspend - JTAG emulator halts chip while FREE = 0 and SOFT = 1 in uPP PERCTL register.

For other settings of FREE and SOFT, the uPP peripheral continues running during emulation halt. When the uPP encounters a stop condition, it completes the current DMA burst transaction (if one is active) before stopping.

I/O channel, configured in transmit mode, immediately places its pins in a high-impedance state and preserves the state of its internal state machines. Unless some reset event occurs (see Section 24.4.5), the channel can resume where it left off when the stop condition is cleared. I/O channel configured in receive mode only captures one additional data word. Further incoming data words are ignored as long as the stop condition persists.

### *24.4.8 Transmit and Receive FIFOs*

Each of the uPP peripheral I/O channels has a 512-byte FIFO. In receive mode, the FIFO is divided into eight 64-byte blocks. In transmit mode, the FIFO is divided into blocks that can be set to 64, 128, or 256 bytes, configured by the TXSIZEA field in the uPP threshold configuration register (THCFG).

Transmission will not begin until the channel has loaded enough data to fill at least one full FIFO block. The internal DMA channels may also be configured to use a read threshold of 64, 128, or 256 bytes using the RDSIZEI or RDSIZEQ field in uPP THCFG register. The DMA write threshold is fixed at 64 bytes.



Copyright © 2024 Texas Instruments Incorporated

### 24.4.9 Transmit and Receive Data (MSG) RAM

On this device, uPP internal DMA does not have access to system memories. Instead, there are two dedicated DATA RAMs (also called as MSGRAM) that have been provided: one for each RX and TX; each of these DATA RAMs are 512B. Since the CPU/CLA operates in 16-bit addressing mode, whereas, the uPP internal DMA operates in byte addressing mode, the addresses for these RAMs are different for different views.

Table 24-2 describes the addresses for these RAMs for two different views.

**Table 24-2. CPU/CLA/uPP-DMA Address Map**

| Data (MSG) RAM | CPU/CLA Address | | uPP DMA (Programming) Address | |
|---|---|---|---|---|
| | Start Address | End Address | Start Address | End Address |
| TX DATA RAM | 0x6C00 | 0x6CFF | 0x6C00 | 0x6DFF |
| RX DATA RAM | 0x6E00 | 0x6EFF | 0x7000 | 0x71FF |

Table 24-3 describes the different access type for both the DATA(MSG) RAMs.

**Table 24-3. CPU/CLA/uPP-DMA Address Map**

| Master/Data RAM | TX Data (MSG) RAM | | RX Data (MSG) RAM | |
|---|---|---|---|---|
| | Read | Write | Read | Write |
| CPU1 | Yes | Yes | Yes | Yes (Debug Mode Only) |
| CPU1.CLA1 | Yes | Yes | Yes | No |
| uPP-DMA | Yes | No | No | Yes |

**Note**

The message RAMs clocks are gated off when uPP clock is gated by the PCLKCR configuration. Thus, the message RAMs are not accessible by CPU/CLA when RPI clock is disabled by CPU.



Copyright © 2024 Texas Instruments Incorporated

### 24.4.10 Initialization and Operation

This section provides step-by-step instructions for initializing and running the uPP peripheral in various modes. These instructions are given assuming that the device has just come out of a power-on reset (POR) state.

1. Apply the appropriate pin multiplexing settings. For more information, see the device-specific data manual, and (or) pin multiplexing utility.
2. Enable the clocks to the uPP peripheral. For more information, see the Clock Control section of this TRM.
3. Wait for few (approximately 32) device clock cycles, and then clear the “Soft Reset” bit to 0 to bring the module out of reset.
4. Program the following uPP configuration registers: CHCTL, IFCFG, IFIVAL, THCFG.
   a. uPP Channel Control Register (data format, SDR/DDR, TX/RX, single/dual channel, interleave, demux)
   b. uPP Interface configuration Register (IO signal enable, polarity, clock divisor)
   c. uPP Interface idle value register (to drive value in idle mode for Tx)
   d. uPP Threshold configuration register (TX Size, DMA read burst size)
5. Program the uPP interrupt enable set register to interrupt generation for the desired events. Register an interrupt service routine (ISR) if desired; otherwise, polling is required.
6. Set the “PerEn” bit in the uPP peripheral control register (PCR) to 1 to turn on the uPP peripheral.
7. Allocate or initialize data buffers for use with uPP.
8. Program the DMA channels with their first transfers using the uPP DMA channel descriptor register0, 1 and 2 (byte count, line count, line offset).
9. Once the DMA descriptors are written, the module starts to receive and transmit data, and performs DMA transfers. Following is a simple description of how the DMA works.
10. Watch for interrupt events. Reprogram the DMA as necessary. (check that the PEND bit in the uPP DMA channel status is 0).
    a. If polling, check uPP Interrupt enabled status register. Reading a bit as 1 indicates the corresponding event has occurred. Write the corresponding bit with 1 to clear.
    b. If using ISR, check ENINTST inside your ISR

Once the channel is started, the initial latency between the time when the first data is received or transmitted on the pin, and the time when the first DMA burst is transferred on the CBA bus, is unpredictable. This initial latency is determined by several factors:

- Clock ratio between system clock and uPP clock
- Various threshold values being programmed
- Chip-level traffic activities level

However, once the data is moving, all subsequent data movement is continuous (streaming). If such data movement cannot be maintained, then DMA under-run or over-flow situation can occur.



Copyright © 2024 Texas Instruments Incorporated

#### 24.4.10.1 System Tuning Tips

The uPP peripheral can operate at high speed and transfer data at a very high rate. When operating the uPP near the upper limits, tuning certain parameters can help decrease the incidence of errors and the software overhead incurred servicing uPP data. Table 24-4 lists several parameters that can be useful in system tuning. A parameter is defined as a “coarse” adjustment, if changing the parameter directly alters the peripheral throughput. A “fine” adjustment does not change the peripheral throughput, but the adjustment does affect general system performance.

**Table 24-4. uPP Parameters Useful for System Tuning**

| Parameter | Register | Register Field | Edge Value | Safe Value | Description |
|---|---|---|---|---|---|
| Data Rate | CHCTL | DRA | 1 | 0 | Double data rate increases data transfer by a factor of 2 and greatly increases system loading for the same clock divisor. This is a coarse adjustment and is probably fixed due to design constraints. |
| Clock Division | IFCFG | CLKDIVA | 0 | 1+ | Increasing clock division is the most straight-forward way to decrease system loading. This is a coarse adjustment; the difference between CLKDIVx = 0 and 1 is the same (in terms of data rate) as the difference between single and double data rate. |
| DMA Read Burst Size | THCFG | RDSIZEQ RDSIZEI | 0 | 3h | Increasing the DMA read threshold decreases system loading by generating fewer, larger DMA events. This is a fine adjustment. |
| DMA Line Size, Count | CHxDESC1 | LCNT BCNT | 0 | (1) | Condensing uPP transfers into fewer, larger lines generates fewer end-of-line interrupts and, thus, invokes fewer ISR calls. This is a fine adjustment. |
| Total Transfer Size | CHxDESC1 | LCNT BCNT | (1) | (1) | Performing many small uPP transfers can require excessive software overhead (programming DMA descriptors, handling interrupts, and so forth) at high data rates. This is a fine adjustment. |

(1) These values vary per application. One example can be a 16-KB transfer. The same total data can be transferred as 16 1-KB lines or two 8-KB lines.

## 24.5 UPP Registers

### *24.5.1 UPP Base Addresses*

**Table 24-5. UPP Base Address Table**

| Device Registers | Register Name | Start Address | End Address |
|---|---|---|---|
| UppRegs(1) | UPP_REGS | 0x0000_6200 | 0x0000_62FF |

(1) Only available on CPU1.



Copyright © 2024 Texas Instruments Incorporated

### 24.5.2 UPP_REGS Registers

Table 24-6 lists the memory-mapped registers for the UPP_REGS registers. All register offset addresses not listed in Table 24-6 should be considered as reserved locations and the register contents should not be modified.

**Table 24-6. UPP_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | PID | Peripheral ID Register | | Go |
| 2h | PERCTL | Peripheral Control Register | | Go |
| 8h | CHCTL | General Control Register | | Go |
| Ah | IFCFG | Interface Configuration Register | | Go |
| Ch | IFIVAL | Interface Idle Value Register | | Go |
| Eh | THCFG | Threshold Configuration Register | | Go |
| 10h | RAWINTST | Raw Interrupt Status Register | | Go |
| 12h | ENINTST | Enable Interrupt Status Register | | Go |
| 14h | INTENSET | Interrupt Enable Set Register | | Go |
| 16h | INTENCLR | Interrupt Enable Clear Register | | Go |
| 20h | CHIDESC0 | DMA Channel I Descriptor 0 Register | | Go |
| 22h | CHIDESC1 | DMA Channel I Descriptor 1 Register | | Go |
| 24h | CHIDESC2 | DMA Channel I Descriptor 2 Register | | Go |
| 28h | CHIST0 | DMA Channel I Status 0 Register | | Go |
| 2Ah | CHIST1 | DMA Channel I Status 1 Register | | Go |
| 2Ch | CHIST2 | DMA Channel I Status 2 Register | | Go |
| 30h | CHQDESC0 | DMA Channel Q Descriptor 0 Register | | Go |
| 32h | CHQDESC1 | DMA Channel Q Descriptor 1 Register | | Go |
| 34h | CHQDESC2 | DMA Channel Q Descriptor 2 Register | | Go |
| 38h | CHQST0 | DMA Channel Q Status 0 Register | | Go |
| 3Ah | CHQST1 | DMA Channel Q Status 1 Register | | Go |
| 3Ch | CHQST2 | DMA Channel Q Status 2 Register | | Go |
| 40h | GINTEN | Global Peripheral Interrupt Enable Register | | Go |
| 42h | GINTFLG | Global Peripheral Interrupt Flag Register | | Go |
| 44h | GINTCLR | Global Peripheral Interrupt Clear Register | | Go |
| 46h | DLYCTL | IO clock data skew control Register | | Go |

Complex bit access types are encoded to fit into small table cells. Table 24-7 shows the codes that are used for access types in this section.

**Table 24-7. UPP_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| R-0 | R<br>-0 | Read<br>Returns 0s |
| Write Type | | |
| W | W | Write |
| W1C | W<br>1C | Write<br>1 to clear |
| W1S | W<br>1S | Write<br>1 to set |
| Reset or Default Value | | |



Copyright © 2024 Texas Instruments Incorporated

**Table 24-7. UPP_REGS Access Type Codes (continued)**

| Access Type | Code | Description |
|---|---|---|
| *-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. |



Copyright © 2024 Texas Instruments Incorporated

#### 24.5.2.1 PID Register (Offset = 0h) [Reset = 44231100h]

PID is shown in Figure 24-17 and described in Table 24-8.

Return to the Summary Table.

Peripheral ID Register

**Figure 24-17. PID Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REVID | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R-44231100h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 24-8. PID Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | REVID | R | 44231100h | Module revision id.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 24.5.2.2 PERCTL Register (Offset = 2h) [Reset = 00000000h]

PERCTL is shown in Figure 24-18 and described in Table 24-9.

Return to the Summary Table.

Peripheral Control Register

**Figure 24-18. PERCTL Register**

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

**Table 24-9. PERCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-8 | RESERVED | R-0 | 0h | Reserved |
| 7 | DMAST | R | 0h | DMA state machine status.<br>0: Idle<br>1: DMA Burst transaction is active.<br>Reset type: CPU1.SYSRSn |
| 6-5 | RESERVED | R-0 | 0h | Reserved |
| 4 | SOFTRST | R/W | 0h | This bit reset all the state machines and certain memory elements inside the RPI module immediately. Software can write this bit to '1', and later write '0' to bring the RPI module out of reset. Note that MMR are NOT reset, except for Interrupt-Raw-Status Register. This reset can be used to recover the RPI from an error condition.<br>To make sure that a graceful or fail-safe reset is performed, software can first disable the 'PerEn' bit, then poll the DMAStatus bit to make sure that all pending VBUSP DMA burst are completed, then perform a software reset.<br>0: De-assert the reset (out of reset).<br>1: Assert the reset (in to reset).<br>Reset type: CPU1.SYSRSn |
| 3 | PEREN | R/W | 0h | This bit can be used to disable or suspend the RPI module. When this bit is programmed to be disabled, the RPI will be stopped (suspended) after all current DMA activity are completed.<br>0: Disable/Suspend the uPP module.<br>1: Enable/Resume the uPP module.<br>Reset type: CPU1.SYSRSn |
| 2 | RTEMU | R/W | 0h | 0: Real Time emulation is disable. Module gets suspended when CPU is supended.<br>1: Real Time emulation is enable.<br>Reset type: CPU1.SYSRSn |
| 1 | SOFT | R/W | 0h | 0: Hard Stop.<br>1: Soft Stop.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 24-9. PERCTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | FREE | R/W | 0h | 0: Software controlled.<br>1: Free Running.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 24.5.2.3 CHCTL Register (Offset = 8h) [Reset = 00000000h]

CHCTL is shown in Figure 24-19 and described in Table 24-10.

Return to the Summary Table.

General Control Register

**Figure 24-19. CHCTL Register**

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

**Table 24-10. CHCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | RESERVED | R-0 | 0h | Reserved |
| 30-17 | RESERVED | R/W | 0h | Reserved |
| 16 | DRA | R/W | 0h | Data rate control.<br>0: Single Data Rate (SDR).<br>1: Double Data Rate (DDR).<br>Reset type: CPU1.SYSRSn |
| 15-5 | RESERVED | R/W | 0h | Reserved |
| 4 | DEMUXA | R/W | 0h | DDR de-multiplexing mode (This bit is only valid for DDR mode):<br>0: De-multiplexing is disable.<br>1: De-multiplexing is Enable (split data into 2 DMA channels).<br>Reset type: CPU1.SYSRSn |
| 3 | SDRTXILA | R/W | 0h | Tx SDR interleave mode (This bit is only valid for SDR mode):<br>0: Interleving is disable.<br>1: Interleving is Enable (split data into 2 DMA channels).<br>Reset type: CPU1.SYSRSn |
| 2 | RESERVED | R/W | 0h | Reserved |
| 1-0 | MODE | R/W | 0h | Operating mode:<br>00: Pure input receive mode.<br>01: Pure output transmit mode.<br>10: Reserved.<br>11: Reserved.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 24.5.2.4 IFCFG Register (Offset = Ah) [Reset = 00000000h]

IFCFG is shown in Figure 24-20 and described in Table 24-11.

Return to the Summary Table.

Interface Configuration Register

**Figure 24-20. IFCFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | RESERVED | | | | | |
| R-0-0h | | R/W-0h | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | RESERVED | | | | | |
| R-0-0h | | R/W-0h | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | TRISENA | CLKINVA | CLKDIVA | | | |
| R-0-0h | | R/W-0h | R/W-0h | R/W-0h | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | WAITA | ENAA | STARTA | WAITPOLA | ENAPOLA | STARTPOLA |
| R-0-0h | | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 24-11. IFCFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | RESERVED | R-0 | 0h | Reserved |
| 29-24 | RESERVED | R/W | 0h | Reserved |
| 23-22 | RESERVED | R-0 | 0h | Reserved |
| 21-16 | RESERVED | R/W | 0h | Reserved |
| 15-14 | RESERVED | R-0 | 0h | Reserved |
| 13 | TRISENA | R/W | 0h | Transmit mode output tri-state control:<br>0: Disable. RPI will not tri-state during idle time and will drive values from the Interface Idle Value Register.<br>1: Enable. RPI will tri-state during idle time.<br>Idle time is the time before and between Window transfers.<br>Reset type: CPU1.SYSRSn |
| 12 | CLKINVA | R/W | 0h | Clock inversion:<br>0: Clock is not inverted<br>1: Clock is inverted<br>When in Rx mode, the clock will be treated as inverted if enabled.<br>When in Tx mode, the clock will be inverted before going out of the pin.<br>Reset type: CPU1.SYSRSn |
| 11-8 | CLKDIVA | R/W | 0h | Clock divider for transmit mode:<br>TX_IOCLK = CHIP_CLK / 2(N+1)<br>Where 'N' is the value programmed into this field.<br>Reset type: CPU1.SYSRSn |
| 7-6 | RESERVED | R-0 | 0h | Reserved |
| 5 | WAITA | R/W | 0h | Enable Usage of WAIT signal:<br>0: Disable (Tx: ignore wait)<br>1: Enable (Tx: honor wait)<br>This bit is only valid for transmit mode, receive mode always drive WAIT signal inactive (except for stop_run situation).<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 24-11. IFCFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 4 | ENAA | R/W | 0h | Enable Usage of ENABLE (WRITE) signal:<br>0: Disable (Rx: ignore enable)<br>1: Enable (Rx: honor enable)<br>This bit is only valid for receive mode, transmit mode always drive ENABLE signal active.<br>Reset type: CPU1.SYSRSn |
| 3 | STARTA | R/W | 0h | Enable Usage of START (SELECT) signal:<br>0: Disable (Rx: ignore start)<br>1: Enable (Rx: honor start)<br>This bit is only valid for receive mode, transmit mode always drive START signal active.<br>Reset type: CPU1.SYSRSn |
| 2 | WAITPOLA | R/W | 0h | Polarity of WAIT signal:<br>0: Active High.<br>1: Active Low.<br>Reset type: CPU1.SYSRSn |
| 1 | ENAPOLA | R/W | 0h | Polarity of ENABLE(WRITE) signal:<br>0: Active High.<br>1: Active Low.<br>Reset type: CPU1.SYSRSn |
| 0 | STARTPOLA | R/W | 0h | Polarity of START(SELECT) signal:<br>0: Active High.<br>1: Active Low.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 24.5.2.5 IFIVAL Register (Offset = Ch) [Reset = 00000000h]

IFIVAL is shown in Figure 24-21 and described in Table 24-12.

Return to the Summary Table.

Interface Idle Value Register

**Figure 24-21. IFIVAL Register**

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

**Table 24-12. IFIVAL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R/W | 0h | Reserved |
| 15-9 | RESERVED | R/W | 0h | Reserved |
| 8-0 | VALA | R/W | 0h | When in transmit mode, this field holds the value that will be driven out when the channel is idle.<br>This includes both situations when TRISEN field of the RPI Interface Configuration Register is enabled, and when wait-state is inserted by the external receiver (WAIT asserted).<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 24.5.2.6 THCFG Register (Offset = Eh) [Reset = 00000000h]

THCFG is shown in Figure 24-22 and described in Table 24-13.

Return to the Summary Table.

Threshold Configuration Register

**Figure 24-22. THCFG Register**

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

**Table 24-13. THCFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-26 | RESERVED | R-0 | 0h | Reserved |
| 25-24 | RESERVED | R/W | 0h | Reserved |
| 23-18 | RESERVED | R-0 | 0h | Reserved |
| 17-16 | TXSIZEA | R/W | 0h | I/O Transmit Threshold:<br>00: 64 Byte<br>01: 128 Byte<br>10: Reserved<br>11: 256 Byte<br>The uPP will hold off transmitting until this threshold is reach in the transmit buffer. The following programming limitation applies:<br>If TX_SIZE = 64B, there is no programming limitation.<br>If TX_SIZE = 128B, descriptor Byte-Count must > 64B.<br>If TX_SIZE = 256B, descriptor Byte-Count must > 192B<br>Reset type: CPU1.SYSRSn |
| 15-10 | RESERVED | R-0 | 0h | Reserved |
| 9-8 | RDSIZEQ | R/W | 0h | DMA Read Threshold for DMA Channel Q:<br>00: 64 Byte<br>01: 128 Byte<br>10: Reserved<br>11: 256 Byte<br>DMA read burst is based on this value (same as FIFO block size).<br>Note: DMA Write Threshold (write burst) is fixed at 64B.<br>Reset type: CPU1.SYSRSn |
| 7-2 | RESERVED | R-0 | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 24-13. THCFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1-0 | RDSIZEI | R/W | 0h | DMA Read Threshold for DMA Channel I:<br>00: 64 Byte<br>01: 128 Byte<br>10: Reserved<br>11: 256 Byte<br>DMA read burst is based on this value (same as FIFO block size).<br>Note: DMA Write Threshold (write burst) is fixed at 64B.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 24.5.2.7 RAWINTST Register (Offset = 10h) [Reset = 00000000h]

RAWINTST is shown in Figure 24-23 and described in Table 24-14.

Return to the Summary Table.

Raw Interrupt Status Register

**Figure 24-23. RAWINTST Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | EOLQ | EOWQ | RESERVED | UOEQ | DPEQ |
| R-0-0h | | | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | EOLI | EOWI | RESERVED | UOEI | DPEI |
| R-0-0h | | | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 24-14. RAWINTST Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-13 | RESERVED | R-0 | 0h | Reserved |
| 12 | EOLQ | R/W | 0h | Interrupt raw status for end-of-line condition:<br>0: No event.<br>1: End Of Line event happened.<br>Reset type: CPU1.SYSRSn |
| 11 | EOWQ | R/W | 0h | Interrupt raw status for end-of-line condition:<br>0: No event.<br>1: End Of Window event happened.<br>Reset type: CPU1.SYSRSn |
| 10 | RESERVED | R/W | 0h | Reserved |
| 9 | UOEQ | R/W | 0h | Interrupt raw status for DMA under-run or over-run :<br>0: No event.<br>1: Under-run/Over-run event happened.<br>Over-run in receiving or Under-Run in transmitting.<br>Reset type: CPU1.SYSRSn |
| 8 | DPEQ | R/W | 0h | Interrupt raw status for DMA programming error:<br>0: No event.<br>1: DMA programming error (Writing of DMA descriptors while PENDING bit is active) occurred.<br>Reset type: CPU1.SYSRSn |
| 7-5 | RESERVED | R-0 | 0h | Reserved |
| 4 | EOLI | R/W | 0h | Interrupt raw status for end-of-line condition:<br>0: No event.<br>1: End Of Line event happened.<br>Reset type: CPU1.SYSRSn |
| 3 | EOWI | R/W | 0h | Interrupt raw status for end-of-line condition:<br>0: No event.<br>1: End Of Window event happened.<br>Reset type: CPU1.SYSRSn |
| 2 | RESERVED | R/W | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 24-14. RAWINTST Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1 | UOEI | R/W | 0h | Interrupt raw status for DMA under-run or over-run :<br>0: No event.<br>1: Under-run/Over-run event happened.<br>Over-run in receiving or Under-Run in transmitting.<br>Reset type: CPU1.SYSRSn |
| 0 | DPEI | R/W | 0h | Interrupt raw status for DMA programming error:<br>0: No event.<br>1: DMA programming error (Writing of DMA descriptors while PENDING bit is active) occurred.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 24.5.2.8 ENINTST Register (Offset = 12h) [Reset = 00000000h]

ENINTST is shown in Figure 24-24 and described in Table 24-15.

Return to the Summary Table.

Enable Interrupt Status Register

**Figure 24-24. ENINTST Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | EOLQ | EOWQ | RESERVED | UOEQ | DPEQ |
| R-0-0h | | | R/W1C-0h | R/W1C-0h | R/W1C-0h | R/W1C-0h | R/W1C-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | EOLI | EOWI | RESERVED | UOEI | DPEI |
| R-0-0h | | | R/W1C-0h | R/W1C-0h | R/W1C-0h | R/W1C-0h | R/W1C-0h |

**Table 24-15. ENINTST Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-13 | RESERVED | R-0 | 0h | Reserved |
| 12 | EOLQ | R/W1C | 0h | Interrupt enable status for end-of-line condition. Writing 1 will clear the interrupt status and writing 0 has no effect.<br>0: No event.<br>1: End Of Line event happened.<br>Reset type: CPU1.SYSRSn |
| 11 | EOWQ | R/W1C | 0h | Interrupt enable status for end-of-line condition. Writing 1 will clear the interrupt status and writing 0 has no effect.<br>0: No event.<br>1: End Of Window event happened.<br>Reset type: CPU1.SYSRSn |
| 10 | RESERVED | R/W1C | 0h | Reserved |
| 9 | UOEQ | R/W1C | 0h | Interrupt enable status for DMA under-run or over-run. Writing 1 will clear the interrupt status and writing 0 has no effect.<br>0: No event.<br>1: Under-run/Over-run event happened.<br>Over-run in receiving or Under-Run in transmitting.<br>Reset type: CPU1.SYSRSn |
| 8 | DPEQ | R/W1C | 0h | Interrupt enable status for DMA programming error. Writing 1 will clear the interrupt status and writing 0 has no effect.<br>0: No event.<br>1: DMA programming error (Writing of DMA descriptors while PENDING bit is active) occurred.<br>Reset type: CPU1.SYSRSn |
| 7-5 | RESERVED | R-0 | 0h | Reserved |
| 4 | EOLI | R/W1C | 0h | Interrupt enable status for end-of-line condition. Writing 1 will clear the interrupt status and writing 0 has no effect.<br>0: No event.<br>1: End Of Line event happened.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 24-15. ENINTST Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3 | EOWI | R/W1C | 0h | Interrupt enable status for end-of-line condition. Writing 1 will clear the interrupt status and writing 0 has no effect.<br>0: No event.<br>1: End Of Window event happened.<br>Reset type: CPU1.SYSRSn |
| 2 | RESERVED | R/W1C | 0h | Reserved |
| 1 | UOEI | R/W1C | 0h | Interrupt enable status for DMA under-run or over-run. Writing 1 will clear the interrupt status and writing 0 has no effect.<br>0: No event.<br>1: Under-run/Over-run event happened.<br>Over-run in receiving or Under-Run in transmitting.<br>Reset type: CPU1.SYSRSn |
| 0 | DPEI | R/W1C | 0h | Interrupt enable status for DMA programming error. Writing 1 will clear the interrupt status and writing 0 has no effect.<br>0: No event.<br>1: DMA programming error (Writing of DMA descriptors while PENDING bit is active) occurred.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 24.5.2.9 INTENSET Register (Offset = 14h) [Reset = 00000000h]

INTENSET is shown in Figure 24-25 and described in Table 24-16.

Return to the Summary Table.

Interrupt Enable Set Register

**Figure 24-25. INTENSET Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | EOLQ | EOWQ | RESERVED | UOEQ | DPEQ |
| R-0-0h | | | R/W1S-0h | R/W1S-0h | R/W1S-0h | R/W1S-0h | R/W1S-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | EOLI | EOWI | RESERVED | UOEI | DPEI |
| R-0-0h | | | R/W1S-0h | R/W1S-0h | R/W1S-0h | R/W1S-0h | R/W1S-0h |

**Table 24-16. INTENSET Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-13 | RESERVED | R-0 | 0h | Reserved |
| 12 | EOLQ | R/W1S | 0h | Interrupt enable for end-of-line condition. Writing 1 will enable the interrupt and writing 0 has no effect.<br>0: Interrupt is disable.<br>1: Interrupt is enable.<br>Reset type: CPU1.SYSRSn |
| 11 | EOWQ | R/W1S | 0h | Interrupt enable for end-of-line condition. Writing 1 will enable the interrupt and writing 0 has no effect.<br>0: Interrupt is disable.<br>1: Interrupt is enable.<br>Reset type: CPU1.SYSRSn |
| 10 | RESERVED | R/W1S | 0h | Reserved |
| 9 | UOEQ | R/W1S | 0h | Interrupt enable for DMA under-run or over-run. Writing 1 will enable the interrupt and writing 0 has no effect.<br>0: Interrupt is disable.<br>1: Interrupt is enable.<br>Reset type: CPU1.SYSRSn |
| 8 | DPEQ | R/W1S | 0h | Interrupt enable for DMA programming error. Writing 1 will enable the interrupt and writing 0 has no effect.<br>0: Interrupt is disable.<br>1: Interrupt is enable.<br>Reset type: CPU1.SYSRSn |
| 7-5 | RESERVED | R-0 | 0h | Reserved |
| 4 | EOLI | R/W1S | 0h | Interrupt enable for end-of-line condition. Writing 1 will enable the interrupt and writing 0 has no effect.<br>0: Interrupt is disable.<br>1: Interrupt is enable.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 24-16. INTENSET Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3 | EOWI | R/W1S | 0h | Interrupt enable for end-of-line condition. Writing 1 will enable the interrupt and writing 0 has no effect.<br>0: Interrupt is disable.<br>1: Interrupt is enable.<br>Reset type: CPU1.SYSRSn |
| 2 | RESERVED | R/W1S | 0h | Reserved |
| 1 | UOEI | R/W1S | 0h | Interrupt enable for DMA under-run or over-run. Writing 1 will enable the interrupt and writing 0 has no effect.<br>0: Interrupt is disable.<br>1: Interrupt is enable.<br>Over-run in receiving or Under-Run in transmitting.<br>Reset type: CPU1.SYSRSn |
| 0 | DPEI | R/W1S | 0h | Interrupt enable for DMA programming error. Writing 1 will enable the interrupt and writing 0 has no effect.<br>0: Interrupt is disable.<br>1: Interrupt is enable.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 24.5.2.10 INTENCLR Register (Offset = 16h) [Reset = 00000000h]

INTENCLR is shown in Figure 24-26 and described in Table 24-17.

Return to the Summary Table.

Interrupt Enable Clear Register

**Figure 24-26. INTENCLR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | EOLQ | EOWQ | RESERVED | UOEQ | DPEQ |
| R-0-0h | | | R/W1S-0h | R/W1S-0h | R/W1S-0h | R/W1S-0h | R/W1S-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | EOLI | EOWI | RESERVED | UOEI | DPEI |
| R-0-0h | | | R/W1S-0h | R/W1S-0h | R/W1S-0h | R/W1S-0h | R/W1S-0h |

**Table 24-17. INTENCLR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-13 | RESERVED | R-0 | 0h | Reserved |
| 12 | EOLQ | R/W1S | 0h | Interrupt clear for end-of-line condition. Writing 1 will disable the interrupt and writing 0 has no effect.<br>0: Interrupt is disable.<br>1: Interrupt is enable.<br>Reset type: CPU1.SYSRSn |
| 11 | EOWQ | R/W1S | 0h | Interrupt clear for end-of-line condition. Writing 1 will disable the interrupt and writing 0 has no effect.<br>0: Interrupt is disable.<br>1: Interrupt is enable.<br>Reset type: CPU1.SYSRSn |
| 10 | RESERVED | R/W1S | 0h | Reserved |
| 9 | UOEQ | R/W1S | 0h | Interrupt clear for DMA under-run or over-run. Writing 1 will disable the interrupt and writing 0 has no effect.<br>0: Interrupt is disable.<br>1: Interrupt is enable.<br>Reset type: CPU1.SYSRSn |
| 8 | DPEQ | R/W1S | 0h | Interrupt clear for DMA programming error. Writing 1 will disable the interrupt and writing 0 has no effect.<br>0: Interrupt is disable.<br>1: Interrupt is enable.<br>Reset type: CPU1.SYSRSn |
| 7-5 | RESERVED | R-0 | 0h | Reserved |
| 4 | EOLI | R/W1S | 0h | Interrupt clear for end-of-line condition. Writing 1 will disable the interrupt and writing 0 has no effect.<br>0: Interrupt is disable.<br>1: Interrupt is enable.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 24-17. INTENCLR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3 | EOWI | R/W1S | 0h | Interrupt clear for end-of-line condition. Writing 1 will disable the interrupt and writing 0 has no effect.<br>0: Interrupt is disable.<br>1: Interrupt is enable.<br>Reset type: CPU1.SYSRSn |
| 2 | RESERVED | R/W1S | 0h | Reserved |
| 1 | UOEI | R/W1S | 0h | Interrupt clear for DMA under-run or over-run. Writing 1 will disable the interrupt and writing 0 has no effect.<br>0: Interrupt is disable.<br>1: Interrupt is enable.<br>Over-run in receiving or Under-Run in transmitting.<br>Reset type: CPU1.SYSRSn |
| 0 | DPEI | R/W1S | 0h | Interrupt clear for DMA programming error. Writing 1 will disable the interrupt and writing 0 has no effect.<br>0: Interrupt is disable.<br>1: Interrupt is enable.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 24.5.2.11 CHIDESC0 Register (Offset = 20h) [Reset = 00000000h]

CHIDESC0 is shown in Figure 24-27 and described in Table 24-18.

Return to the Summary Table.

DMA Channel I Descriptor 0 Register

**Figure 24-27. CHIDESC0 Register**

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

**Table 24-18. CHIDESC0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | ADDR | R/W | 0h | Starting address of the DMA Channel I transfer. It must be 64bit aligned.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 24.5.2.12 CHIDESC1 Register (Offset = 22h) [Reset = 00000000h]

CHIDESC1 is shown in Figure 24-28 and described in Table 24-19.

Return to the Summary Table.

DMA Channel I Descriptor 1 Register

**Figure 24-28. CHIDESC1 Register**

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

**Table 24-19. CHIDESC1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | LCNT | R/W | 0h | Number of lines in a window for DMA Channel I transfer(number of packets in CPPI 4.1 terminology):<br>0: Line count of 0 (invalid programming)<br>1: Line count of 1<br>So on...<br>Reset type: CPU1.SYSRSn |
| 15-0 | BCNT | R/W | 0h | Number of bytes in a line for DMA Channel I transfer(number of bytes in a packet in CPPI 4.1 terminology):<br>0: Byte count of 0 (invalid programming)<br>1: Byte count of 1<br>So on...<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 24.5.2.13 CHIDESC2 Register (Offset = 24h) [Reset = 00000000h]

CHIDESC2 is shown in Figure 24-29 and described in Table 24-20.

Return to the Summary Table.

DMA Channel I Descriptor 2 Register

**Figure 24-29. CHIDESC2 Register**

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

**Table 24-20. CHIDESC2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-0 | LOFFSET | R/W | 0h | Offset from the current line starting address to the next line starting address for DMA Channel I transfers. It must be 64bit aligned.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 24.5.2.14 CHIST0 Register (Offset = 28h) [Reset = 00000000h]

CHIST0 is shown in Figure 24-30 and described in Table 24-21.

Return to the Summary Table.

DMA Channel I Status 0 Register

**Figure 24-30. CHIST0 Register**

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

**Table 24-21. CHIST0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | ADDR | R | 0h | Current address of the DMA transfer.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 24.5.2.15 CHIST1 Register (Offset = 2Ah) [Reset = 00000000h]

CHIST1 is shown in Figure 24-31 and described in Table 24-22.

Return to the Summary Table.

DMA Channel I Status 1 Register

**Figure 24-31. CHIST1 Register**

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

**Table 24-22. CHIST1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | LCNT | R | 0h | Current line number.<br>Reset type: CPU1.SYSRSn |
| 15-0 | BCNT | R | 0h | Current byte number.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 24.5.2.16 CHIST2 Register (Offset = 2Ch) [Reset = 00000000h]

CHIST2 is shown in Figure 24-32 and described in Table 24-23.

Return to the Summary Table.

DMA Channel I Status 2 Register

**Figure 24-32. CHIST2 Register**

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

**Table 24-23. CHIST2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-8 | RESERVED | R-0 | 0h | Reserved |
| 7-4 | WM | R | 0h | Watermark for FIFO block count for DMA Channel I tranfer.<br>For RX, this is a recording of the maximum FIFO Block Occupancy ever reached for a continuous transaction.<br>For TX, this is a simple capture of the FIFO Block Emptiness count every clock.<br>Reset type: CPU1.SYSRSn |
| 3-2 | RESERVED | R-0 | 0h | Reserved |
| 1 | PEND | R | 0h | Status of DMA:<br>0: OK to write a new set of DMA descriptor.<br>1: Writing of new DMA descriptor is disallowed/ignored.<br>Reset type: CPU1.SYSRSn |
| 0 | ACT | R | 0h | Status of DMA descriptor.t:<br>0: Descriptor is currently idle.<br>1: Descriptor is currently active (transferring data).<br>'PENDING' bit is used for descriptor programming allowance, while 'ACTIVE' is used for indicating if a descriptor is being in use or not. Software should not use these bit to indicate 'end-of-window' transfer condition (use the EOW status/interrupt instead).<br>For RX mode, this bit reflects if any of the 2 pending descriptors on the CBA clock domain is currently running.<br>For TX mode, this bit reflects if any of the 2 pending descriptors on the RPI clock domain is currently running.<br>A proper way to monitor this signal (after loading a descriptor) is to first wait for this ACTIVE signal to go active, then wait for this signal to go inactive. The deassertion of the ACTIVE signal indicates that the descriptor is completed.<br>Another side note is that the 'DMA_STATUS' of PCR is a low-level monitoring signal for DMA VBUSP bus activity, while the PENDING, ACTIVE are higher-lever monitoring for descriptor status.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 24.5.2.17 CHQDESC0 Register (Offset = 30h) [Reset = 00000000h]

CHQDESC0 is shown in Figure 24-33 and described in Table 24-24.

Return to the Summary Table.

DMA Channel Q Descriptor 0 Register

**Figure 24-33. CHQDESC0 Register**

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

**Table 24-24. CHQDESC0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | ADDR | R/W | 0h | Starting address of the DMA Channel Q transfer. This must be 64bit aligned.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 24.5.2.18 CHQDESC1 Register (Offset = 32h) [Reset = 00000000h]

CHQDESC1 is shown in Figure 24-34 and described in Table 24-25.

Return to the Summary Table.

DMA Channel Q Descriptor 1 Register

**Figure 24-34. CHQDESC1 Register**

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

**Table 24-25. CHQDESC1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | LCNT | R/W | 0h | Number of lines in a window for DMA Channel Q transfer(number of packets in CPPI 4.1 terminology):<br>0: Line count of 0 (invalid programming)<br>1: Line count of 1<br>So on...<br>Reset type: CPU1.SYSRSn |
| 15-0 | BCNT | R/W | 0h | Number of bytes in a line for DMA Channel Q transfer(number of bytes in a packet in CPPI 4.1 terminology):<br>0: Byte count of 0 (invalid programming)<br>1: Byte count of 1<br>So on...<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 24.5.2.19 CHQDESC2 Register (Offset = 34h) [Reset = 00000000h]

CHQDESC2 is shown in Figure 24-35 and described in Table 24-26.

Return to the Summary Table.

DMA Channel Q Descriptor 2 Register

**Figure 24-35. CHQDESC2 Register**

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

**Table 24-26. CHQDESC2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-0 | LOFFSET | R/W | 0h | Offset from the current line starting address to the next line starting address for DMA Channel Q transfers. It must be 64bit aligned.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 24.5.2.20 CHQST0 Register (Offset = 38h) [Reset = 00000000h]

CHQST0 is shown in Figure 24-36 and described in Table 24-27.

Return to the Summary Table.

DMA Channel Q Status 0 Register

**Figure 24-36. CHQST0 Register**

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

**Table 24-27. CHQST0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | ADDR | R | 0h | Current address of the DMA transfer.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 24.5.2.21 CHQST1 Register (Offset = 3Ah) [Reset = 00000000h]

CHQST1 is shown in Figure 24-37 and described in Table 24-28.

Return to the Summary Table.

DMA Channel Q Status 1 Register

**Figure 24-37. CHQST1 Register**

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

**Table 24-28. CHQST1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | LCNT | R | 0h | Current line number.<br>Reset type: CPU1.SYSRSn |
| 15-0 | BCNT | R | 0h | Current byte number.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 24.5.2.22 CHQST2 Register (Offset = 3Ch) [Reset = 00000000h]

CHQST2 is shown in Figure 24-38 and described in Table 24-29.

Return to the Summary Table.

DMA Channel Q Status 2 Register

**Figure 24-38. CHQST2 Register**

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

**Table 24-29. CHQST2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-8 | RESERVED | R-0 | 0h | Reserved |
| 7-4 | WM | R | 0h | Watermark for FIFO block count for DMA Channel Q tranfer.<br>For RX, this is a recording of the maximum FIFO Block Occupancy ever reached for a continuous transaction.<br>For TX, this is a simple capture of the FIFO Block Emptiness count every clock.<br>Reset type: CPU1.SYSRSn |
| 3-2 | RESERVED | R-0 | 0h | Reserved |
| 1 | PEND | R | 0h | Status of DMA:<br>0: OK to write a new set of DMA descriptor.<br>1: Writing of new DMA descriptor is disallowed/ignored.<br>Reset type: CPU1.SYSRSn |
| 0 | ACT | R | 0h | Status of DMA descriptor.t:<br>0: Descriptor is currently idle.<br>1: Descriptor is currently active (transferring data).<br>'PENDING' bit is used for descriptor programming allowance, while 'ACTIVE' is used for indicating if a descriptor is being in use or not. Software should not use these bit to indicate 'end-of-window' transfer condition (use the EOW status/interrupt instead).<br>For RX mode, this bit reflects if any of the 2 pending descriptors on the CBA clock domain is currently running.<br>For TX mode, this bit reflects if any of the 2 pending descriptors on the RPI clock domain is currently running.<br>A proper way to monitor this signal (after loading a descriptor) is to first wait for this ACTIVE signal to go active, then wait for this signal to go inactive. The deassertion of the ACTIVE signal indicates that the descriptor is completed.<br>Another side note is that the 'DMA_STATUS' of PCR is a low-level monitoring signal for DMA VBUSP bus activity, while the PENDING, ACTIVE are higher-lever monitoring for descriptor status.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 24.5.2.23 GINTEN Register (Offset = 40h) [Reset = 00000000h]

GINTEN is shown in Figure 24-39 and described in Table 24-30.

Return to the Summary Table.

Global Peripheral Interrupt Enable Register

**Figure 24-39. GINTEN Register**

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

**Table 24-30. GINTEN Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-1 | RESERVED | R-0 | 0h | Reserved |
| 0 | GINTEN | R/W | 0h | 0 = uPP does not generate interrupt.<br>1= uPP generates interrupt to if interrupt flag gets set.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 24.5.2.24 GINTFLG Register (Offset = 42h) [Reset = 00000000h]

GINTFLG is shown in Figure 24-40 and described in Table 24-31.

Return to the Summary Table.

Global Peripheral Interrupt Flag Register

**Figure 24-40. GINTFLG Register**

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

**Table 24-31. GINTFLG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-1 | RESERVED | R-0 | 0h | Reserved |
| 0 | GINTFLG | R | 0h | 0: No interrupt has been generated.<br>1: Interrupt has been generated.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 24.5.2.25 GINTCLR Register (Offset = 44h) [Reset = 00000000h]

GINTCLR is shown in Figure 24-41 and described in Table 24-32.

Return to the Summary Table.

Global Peripheral Interrupt Clear Register

**Figure 24-41. GINTCLR Register**

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

**Table 24-32. GINTCLR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-1 | RESERVED | R-0 | 0h | Reserved |
| 0 | GINTCLR | R-0/W1S | 0h | Write '1' to this clears the flag in GINTFR. Read always returns '0'.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

#### 24.5.2.26 DLYCTL Register (Offset = 46h) [Reset = 00000001h]

DLYCTL is shown in Figure 24-42 and described in Table 24-33.

Return to the Summary Table.

IO clock data skew control Register

**Figure 24-42. DLYCTL Register**

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

**Table 24-33. DLYCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-3 | RESERVED | R-0 | 0h | Reserved |
| 2-1 | DLYCTL | R/W | 0h | Controls the delay on the input signals for uPP.<br>00: Data and control pins have 4 cycle dealy, clock pins have 2 cycle delay.<br>01: Data and control pins have 6 cycle dealy, clock pins have 2 cycle delay.<br>10: Data and control pins have 9 cycle dealy, clock pins have 2 cycle delay.<br>11: Data and control pins have 14 cycle dealy, clock pins have 2 cycle delay.<br>Reset type: CPU1.SYSRSn |
| 0 | DLYDIS | R/W | 1h | 0: Dealy on pins are controlled by setting in DLYCTL field.<br>1: No extra dealy on pins.<br>Reset type: CPU1.SYSRSn |

### *24.5.3 UPP Registers to Driverlib Functions*

**Table 24-34. UPP Registers to Driverlib Functions**

| File | Driverlib Function |
|---|---|
| **PID** | |
| - | |
| **PERCTL** | |
| upp.h | UPP_isDMAActive |
| upp.h | UPP_performSoftReset |
| upp.h | UPP_enableModule |
| upp.h | UPP_disableModule |
| upp.h | UPP_enableEmulationMode |
| upp.h | UPP_disableEmulationMode |
| upp.h | UPP_setEmulationMode |



Copyright © 2024 Texas Instruments Incorporated

**Table 24-34. UPP Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| **CHCTL** | |
| upp.h | UPP_setOperationMode |
| upp.h | UPP_setDataRate |
| upp.h | UPP_setTxSDRInterleaveMode |
| upp.h | UPP_setDDRDemuxMode |
| **IFCFG** | |
| upp.h | UPP_setControlSignalPolarity |
| upp.h | UPP_setTxControlSignalMode |
| upp.h | UPP_setRxControlSignalMode |
| upp.h | UPP_setTxClockDivider |
| upp.h | UPP_setClockPolarity |
| upp.h | UPP_configTxIdleDataMode |
| **IFIVAL** | |
| upp.h | UPP_setTxIdleValue |
| **THCFG** | |
| upp.c | UPP_setDMAReadThreshold |
| upp.h | UPP_setTxThreshold |
| **RAWINTST** | |
| upp.h | UPP_getRawInterruptStatus |
| **ENINTST** | |
| upp.h | UPP_getInterruptStatus |
| upp.h | UPP_clearInterruptStatus |
| **INTENSET** | |
| upp.h | UPP_enableInterrupt |
| **INTENCLR** | |
| upp.h | UPP_disableInterrupt |
| **CHIDESC0** | |
| upp.c | UPP_setDMADescriptor |
| **CHIDESC1** | |
| upp.c | UPP_setDMADescriptor |
| **CHIDESC2** | |
| upp.c | UPP_setDMADescriptor |
| **CHIST0** | |
| upp.c | UPP_getDMAChannelStatus |
| **CHIST1** | |
| upp.c | UPP_getDMAChannelStatus |
| **CHIST2** | |
| upp.c | UPP_isDescriptorPending |
| upp.c | UPP_isDescriptorActive |
| upp.c | UPP_getDMAFIFOWatermark |
| **CHQDESC0** | |
| upp.c | UPP_setDMADescriptor |
| **CHQDESC1** | |
| upp.c | UPP_setDMADescriptor |
| **CHQDESC2** | |



Copyright © 2024 Texas Instruments Incorporated

**Table 24-34. UPP Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| upp.c | UPP_setDMADescriptor |
| **CHQST0** | |
| upp.c | UPP_getDMAChannelStatus |
| **CHQST1** | |
| upp.c | UPP_getDMAChannelStatus |
| **CHQST2** | |
| upp.c | UPP_isDescriptorPending |
| upp.c | UPP_isDescriptorActive |
| upp.c | UPP_getDMAFIFOWatermark |
| **GINTEN** | |
| upp.h | UPP_enableGlobalInterrupt |
| upp.h | UPP_disableGlobalInterrupt |
| **GINTFLG** | |
| upp.h | UPP_isInterruptGenerated |
| **GINTCLR** | |
| upp.h | UPP_clearGlobalInterruptStatus |
| **DLYCTL** | |
| upp.h | UPP_enableInputDelay |
| upp.h | UPP_disableInputDelay |
| upp.h | UPP_setInputDelay |



Copyright © 2024 Texas Instruments Incorporated

Chapter 25

# External Memory Interface (EMIF)

This chapter describes the external memory interface (EMIF).

Further information can be found in the following documents:

*Accessing External SDRAM on the TMS320F2837x/2807x Microcontrollers Using C/C++ Application Report*

*Design and Usage Guidelines for the C2000™ External Memory Interface (EMIF) Application Report*


**25.1 Introduction**.....2802
**25.2 EMIF Module Architecture**.....2806
**25.3 Example Configuration**.....2836
**25.4 EMIF Registers**.....2844





Copyright © 2024 Texas Instruments Incorporated

## 25.1 Introduction

This device supports dual-core architecture. To have a dedicated EMIF for each CPU subsystem, the device supports two EMIF modules — EMIF1 and EMIF2, as shown in Figure 25-1. Both modules are exactly the same with the same feature set, but have different address/data sizes. EMIF1 is shared between the CPU1 and CPU2 subsystem, whereas EMIF2 is dedicated to the CPU1 subsystem.

Table 25-1 gives the configuration for two EMIF modules.

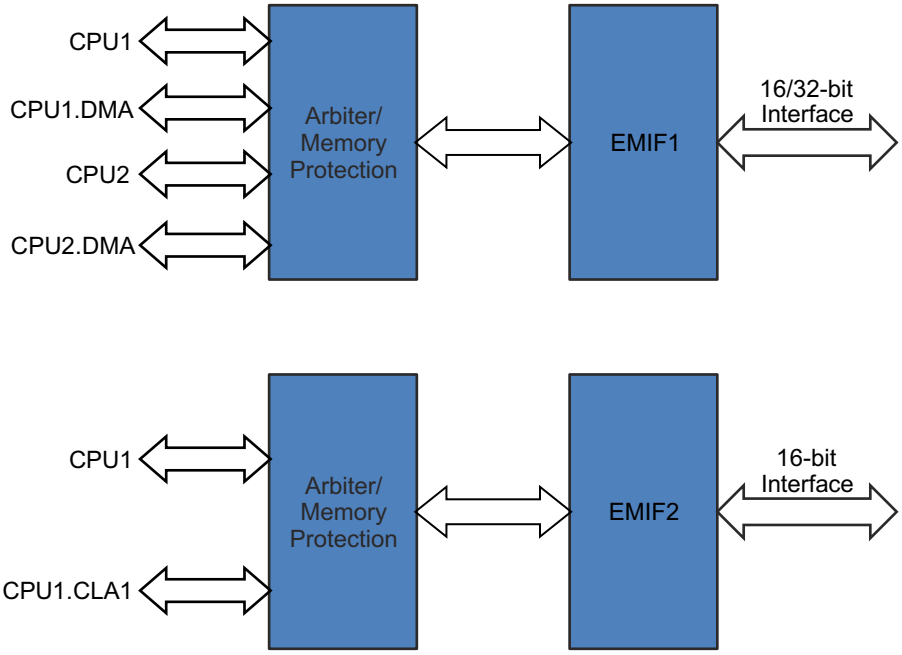



**Figure 25-1. EMIF Module Overview**

**Table 25-1. Configuration for EMIF1 and EMIF2 Modules**

| | EMIF1 | EMIF2 |
|---|---|---|
| 176-Pin Package | Yes | NA |
| 337- Pin Package | Yes | Yes |
| Maximum Data Width | 32 | 16 |
| Maximum Address Width | 22 (Some of the EMIF1 pins are muxed with each other. Refer to Section 25.2.11 for usage) | 12 |
| SDRAM CSx Support | 1 (CS0) | 1 (CS0) |
| ASRAM CSx Support | 3 (CS2/CS3/CS4) | 1 (CS2) |

**Note**

Subsequent sections in this chapter provide the details on the generic EMIF module. Unless otherwise specified, pin names are used from EMIF1 to define the functionality.

On this device, if EMIF1 is chosen to have a 32-bit data width, EMIF2 cannot be used because EMIF2 data pins are muxed with EMIF1 MSB data pins.



Copyright © 2024 Texas Instruments Incorporated

### 25.1.1 Purpose of the Peripheral

This EMIF memory controller is compliant with the JESD21-C SDR SDRAM memories utilizing a 32-bit/16-bit data bus. The purpose of this EMIF is to provide a means for the CPU to connect to a variety of external devices including:

- Single data rate (SDR) SDRAM
- Asynchronous devices including NOR Flash and SRAM

A common use for the EMIF is to interface with both a Flash device and an SDRAM device simultaneously. Section 25.3 contains an example of operating the EMIF in this configuration.

### 25.1.2 EMIF Related Collateral

**Foundational Materials**

- C2000 Academy - EMIF

**Getting Started Materials**

- *Design and Usage Guidelines for the C2000 External Memory Interface (EMIF) Application Report*

**Expert Materials**

- *Accessing External SDRAM on the TMS320F2837x/2807x Microcontrollers Using C/C++ Application Report*
  - In addition to the devices in the title, this material also applies to other devices with EMIF.



Copyright © 2024 Texas Instruments Incorporated

### 25.1.3 Features

The EMIF controller includes many features to enhance the ease and flexibility of connecting to the external SDR SDRAM and asynchronous devices.

#### 25.1.3.1 Asynchronous Memory Support

The EMIF controller supports asynchronous:

- SRAM memories
- NOR Flash memories

There is an external wait input that allows slower asynchronous memories to extend the memory access. The EMIF module supports more than one chip select (enable). Each chip select has the following individually programmable attributes:

- Data bus width
- Read cycle timings: setup, hold, strobe
- Write cycle timings: setup, hold, strobe
- Bus turnaround time
- Extended wait option with programmable timeout
- Select strobe option

#### 25.1.3.2 Synchronous DRAM Memory Support

The EMIF module supports 16-bit/32-bit SDRAM in addition to the asynchronous memories listed in Section 25.1.3.1. The EMIF module has a single SDRAM chip select. SDRAM configurations that are supported are:

- One, two and four bank SDRAM devices
- Devices with eight, nine, ten, and eleven column address
- CAS latency of two or three clock cycles
- 16-bit/32-bit data bus width
- 3.3V LVCMOS interface

Additionally, the EMIF supports placing the SDRAM in self-refresh and power-down modes. The self-refresh mode allows the SDRAM to be put in a low-power state while still retaining memory contents, since the SDRAM continues to refresh itself even without clocks from the microcontroller. The power-down mode achieves even lower power, except the microcontroller must periodically wake up and issue refreshes if data retention is required.

Note that the EMIF module does not support mobile SDRAM devices.



Copyright © 2024 Texas Instruments Incorporated

### 25.1.4 Functional Block Diagram

Figure 25-2 shows the connections between the EMIF and the internal requesters, along with the external EMIF pins. Section 25.2.2 contains a description of the entities internal to the MCU that can send requests to the EMIF, along with their prioritization. Section 25.2.3 describes the EMIF external pins and summarizes their purpose when interfacing with the SDRAM and asynchronous devices.

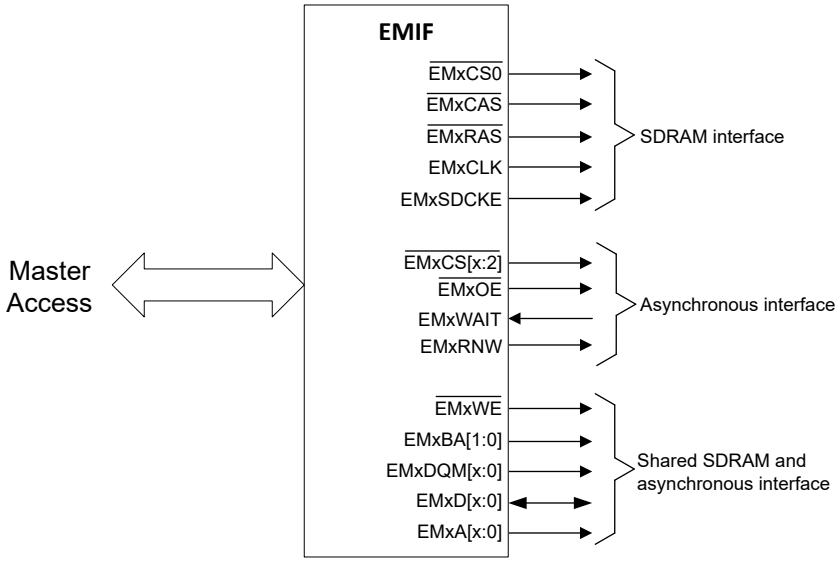



**Figure 25-2. EMIF Functional Block Diagram**

### 25.1.5 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.



Copyright © 2024 Texas Instruments Incorporated

## 25.2 EMIF Module Architecture

This section provides details about the architecture and operation of the EMIF. Both the SDRAM and asynchronous interface are covered, along with other system-related configurations such as clock control.

### *25.2.1 EMIF Clock Control*

The EMIF clock is output on the EMxCLK pin and must be used when interfacing to external SDRAM devices. The EMIF module gets the PLLSYSCLK clock domain as the input. The user can choose to run the EMIF at PLLSYSCLK/1 or PLLSYSCLK/2 clock frequency by configuring the EMIFxCLKDIV field in the PERCLKDIVSEL register in the *Clock Control* module.

### *25.2.2 EMIF Requests*

Different sources within the MCU can make requests to the EMIF. These requests consist of accesses to the SDRAM memory, the asynchronous memory, and the EMIF registers. The EMIF can process only one request at a time. Therefore, a high performance master arbitration block exists within the MCU to provide prioritized requests from the different sources to the EMIF. The sources are:

- CPU1
- CPU1.DMA
- CPU2
- CPU2.DMA

If a request is submitted from two or more sources simultaneously, the crossbar switch forwards the highest priority request to the EMIF first. Upon completion of a request, the master arbitration block again evaluates the pending requests and forwards the highest priority pending request to the EMIF.

The master arbitration block always allows RD access from any of the masters. But for WR access (or execute access), the master arbitration block only allows access of masters from a CPU subsystem that takes master ownership of the EMIF module based on the configuration in the EMIF1MSEL register in the *Memory Controller* module. Note that only the EMIF1 has access from both the CPU subsystems; hence, the concept of grab-semaphore is applicable for the EMIF1 only.

When the EMIF receives a request, it is possible that the request is not immediately processed. In some cases, the EMIF performs one or more auto-refresh cycles before processing the request. For details on the EMIF internal arbitration between performing requests and performing auto-refresh cycles, see Section 25.2.13.



Copyright © 2024 Texas Instruments Incorporated

### 25.2.3 EMIF Signal Descriptions

This section describes the function of each of the EMIF signals.

**Table 25-2. EMIF Pins Used to Access Both SDRAM and Asynchronous Memories**

| Pins | I/O | Description |
|---|---|---|
| EM1D[x:0] | I/O | **EMIF data bus.** |
| EM1A[x:0] | O | **EMIF address bus.**<br>When interfacing to an SDRAM device, these pins are primarily used to provide the row and column address to the SDRAM. The mapping from the internal program address to the external values placed on these pins is found in Table 25-14. EM1A[10] is also used during the PRE command to select which banks to deactivate.<br>When interfacing to an asynchronous device, these pins are used in conjunction with the EM1BA pins to form the address that is sent to the device. The mapping from the internal program address to the external values placed on these pins is found in Section 25.2.6.1. |
| EM1BA[1:0] | O | **EMIF bank address.**<br>When interfacing to an SDRAM device, these pins are used to provide the bank address inputs to the SDRAM. The mapping from the internal program address to the external values placed on these pins is found in Table 25-14.<br>When interfacing to an asynchronous device, these pins are used in conjunction with the EM1A pins to form the address that is sent to the device. The mapping from the internal program address to the external values placed on these pins is found in Section 25.2.6.1. |
| EM1DQM[x:0] | O | **Active-low byte enables.**<br>When interfacing to SDRAM, these pins are connected to the DQM pins of the SDRAM to individually enable/disable each of the bytes in a data access.<br>When interfacing to an asynchronous device, these pins are connected to byte enables. See Section 25.2.6 for details. |
| $\overline{\text{EM1WE}}$ | O | **Active-low write enable.**<br>When interfacing to SDRAM, this pin is connected to the nWE pin of the SDRAM and is used to send commands to the device.<br>When interfacing to an asynchronous device, this pin provides a signal which is active-low during the strobe period of an asynchronous write access cycle. |

**Table 25-3. EMIF Pins Specific to SDRAM**

| Pins | I/O | Description |
|---|---|---|
| $\overline{\text{EM1CS0}}$ | O | **Active-low chip enable pin for SDRAM devices.**<br>This pin is connected to the chip-select pin of the attached SDRAM device and is used for enabling/disabling commands. By default, EMIF keeps this SDRAM chip select active, even if EMIF is not interfaced with an SDRAM device. This pin is deactivated when accessing the asynchronous memory bank and is reactivated on completion of the asynchronous access. |
| EM1RAS | O | **Active-low row address strobe pin.**<br>This pin is connected to the nRAS pin of the attached SDRAM device and is used for sending commands to the device. |
| EM1CAS | O | **Active-low column address strobe pin.**<br>This pin is connected to the nCAS pin of the attached SDRAM device and is used for sending commands to the device. |
| EM1SDCKE | O | **Clock enable pin.**<br>This pin is connected to the CKE pin of the attached SDRAM device and is used for issuing the SELF REFRESH command which places the device in self-refresh mode. See Section 25.2.5.7 for details. |
| EM1CLK | O | **SDRAM clock pin.**<br>This pin is connected to the CLK pin of the attached SDRAM device. See Section 25.2.1 for details on the clock signal. |



Copyright © 2024 Texas Instruments Incorporated

**Table 25-4. EMIF Pins Specific to Asynchronous Memory**

| Pins | I/O | Description |
|---|---|---|
| $\overline{\text{EM1CS}}$[4:2] | O | **Active-low chip enable pins for asynchronous devices.** These pins are meant to be connected to the chip-select pins of the attached asynchronous device. These pins are active only during accesses to the asynchronous memory. |
| EM1WAIT | I | **Wait input with programmable polarity / NAND Flash ready input.** A connected asynchronous device can extend the strobe period of an access cycle by asserting the EM1WAIT input to EMIF as described in Section 25.2.6.6. To enable this functionality, the EW bit in the asynchronous 1 configuration register (ASYNC_CS2_CFG) must be set to 1. In addition, the WP0 bit in ASYNC_CS2_CFG must be configured to define the polarity of the EM1WAIT pin. When the CS2NAND/CS3NAND/CS4NAND/CS5NAND bit in the NAND Flash control register (NANDFCR) is set, this pin instead functions as a NAND Flash ready input. |
| $\overline{\text{EM1OE}}$ | O | **Active-low pin enable for asynchronous devices.** This pin provides a signal which is active-low during the strobe period of an asynchronous read access cycle. |
| EM1RNW | O | **EMIF asynchronous read/write control.** This pin stays high during reads and stays low during writes (same duration as CS). |

### *25.2.4 EMIF Signal Multiplexing Control*

Several EMIF signals are multiplexed with other functions on this microcontroller. Refer to the multiplexing section of the *General-Purpose Input/Output (GPIO)* chapter for more information on how to enable the output of these EMIF signals.

### *25.2.5 SDRAM Controller and Interface*

The EMIF controller provides a glueless interface to most standard SDR SDRAM devices and supports features like self-refresh mode and prioritized refresh. In addition, it provides flexibility through programmable parameters such as the refresh rate, CAS latency, and many SDRAM timing parameters. The following sections include details on how to Interface and properly configure the EMIF to perform read and write operations to externally connected SDR SDRAM devices. Also, Section 25.3 provides a detailed example of interfacing the EMIF to a common SDRAM device.

#### 25.2.5.1 SDRAM Commands

The EMIF controller supports the SDRAM commands described in Table 25-5. Table 25-6 shows the truth table for the SDRAM commands, and an example timing waveform of the PRE command is shown in Figure 25-3. EM1A[10] is pulled low in this example to deactivate only the bank specified by the EM1BA pins.

**Table 25-5. EMIF SDRAM Commands**

| Command | Function |
|---|---|
| PRE | **Precharge.** Depending on the value of EM1A[10], the PRE command either deactivates the open row in all banks (EM1A[10] = 1) or only the bank specified by the EM1BA[1:0] pins (EM1A[10] = 0). |
| ACTV | **Activate**. The ACTV command activates the selected row in a particular bank for the current access. |
| READ | **Read.** The READ command outputs the starting column address and signals the SDRAM to begin the burst read operation. Address EM1A[10] is always pulled low to avoid auto precharge. This allows for better bank interleaving performance. |
| WRT | **Write.** The WRT command outputs the starting column address and signals the SDRAM to begin the burst write operation. Address EM1A[10] is always pulled low to avoid auto precharge. This allows for better bank interleaving performance. |
| BT | **Burst terminate**. The BT command is used to truncate the current read or write burst request. On this device, all the SDRAM accesses are single access except when EMIF controller splits a single access into multiple access (for example, a 32-bit access from CPU is split into two 16-bit accesses if external SDRAM device is 16 bit (NM =1). |
| LMR | **Load mode register**. The LMR command sets the mode register of the attached SDRAM devices and is only issued during the SDRAM initialization sequence described in Section 25.2.5.4. |
| REFR | **Auto refresh**. The REFR command signals the SDRAM to perform an auto refresh according to the internal address. |
| SLFR | **Self refresh**. The self-refresh command places the SDRAM into self-refresh mode, during which the SDRAM provides a clock signal and auto refresh cycles. |
| NOP | **No operation**. The NOP command is issued during all cycles in which one of the above commands is not issued. |



Copyright © 2024 Texas Instruments Incorporated

**Table 25-6. Truth Table for SDRAM Commands**

| SDRAM Pins: | CKE | nCS | nRAS | nCAS | nWE | BA[1:0] | A[12:11] | A[10] | A[9:0] |
|---|---|---|---|---|---|---|---|---|---|
| EMIF Pins: | EM1SDCKE | $\overline{\text{EM1CS}}$[0] | EM1RAS | EM1CAS | $\overline{\text{EM1WE}}$ | EM1BA[1:0] | EM1A[12:11] | EM1A[10] | EM1A[9:0] |
| PRE | H | L | L | H | L | Bank/X | X | L/H | X |
| ACTV | H | L | L | H | H | Bank | Row | Row | Row |
| READ | H | L | H | L | H | Bank | Column | L | Column |
| WRT | H | L | H | L | L | Bank | Column | L | Column |
| BT | H | L | H | H | L | X | X | X | X |
| LMR | H | L | L | L | L | X | Mode | Mode | Mode |
| REFR | H | L | L | L | H | X | X | X | X |
| SLFR | L | L | L | L | H | X | X | X | X |
| NOP | H | L | H | H | H | X | X | X | X |

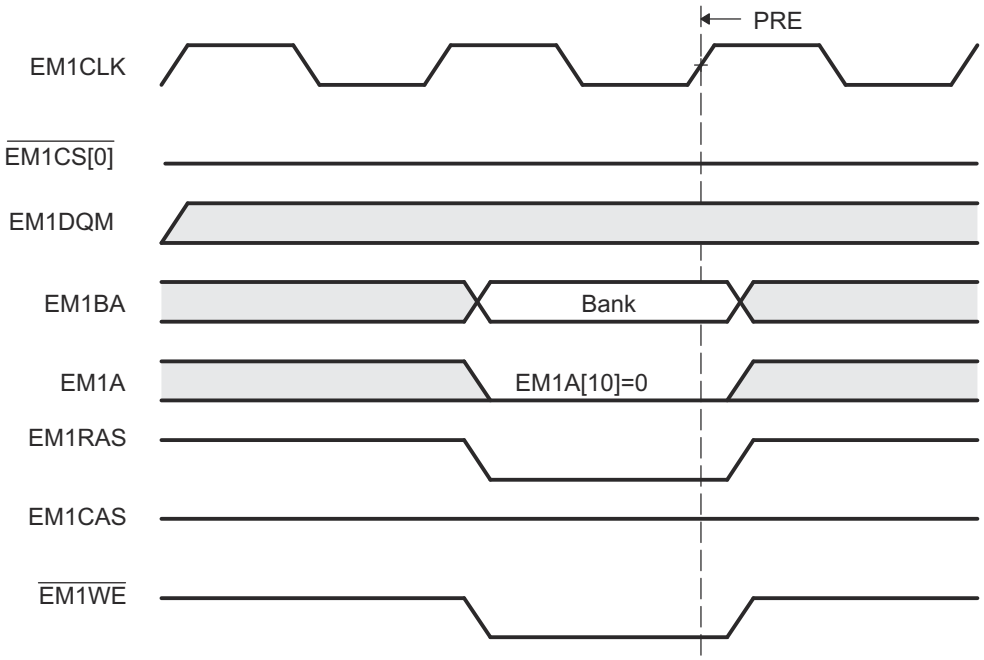



**Figure 25-3. Timing Waveform of SDRAM PRE Command**



Copyright © 2024 Texas Instruments Incorporated

#### 25.2.5.2 Interfacing to SDRAM

The EMIF supports a glueless interface to SDRAM devices with the following characteristics:

- Pre-charge bit is A[10]
- The number of column address bits is 8, 9, 10, or 11.
- The number of row address bits is 13, 14, 15, or 16.
- The number of internal banks is 1, 2, or 4.

Figure 25-4 shows an interface between the EMIF and a 2M × 16 × 4 bank SDRAM device, and Figure 25-5 shows an interface between the EMIF and a 512K × 16 × 2 bank SDRAM device. For devices supporting 16-bit interface, refer to Table 25-7 for list of commonly-supported SDRAM devices and the required connections for the address pins.

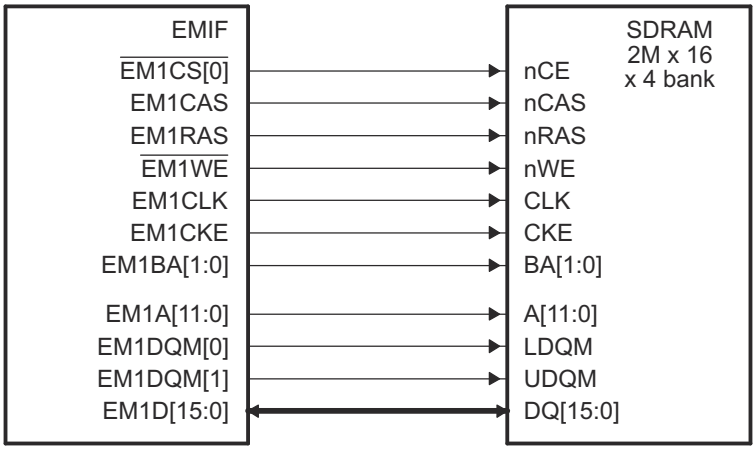



**Figure 25-4. EMIF to 2M × 16 × 4 Bank SDRAM Interface**

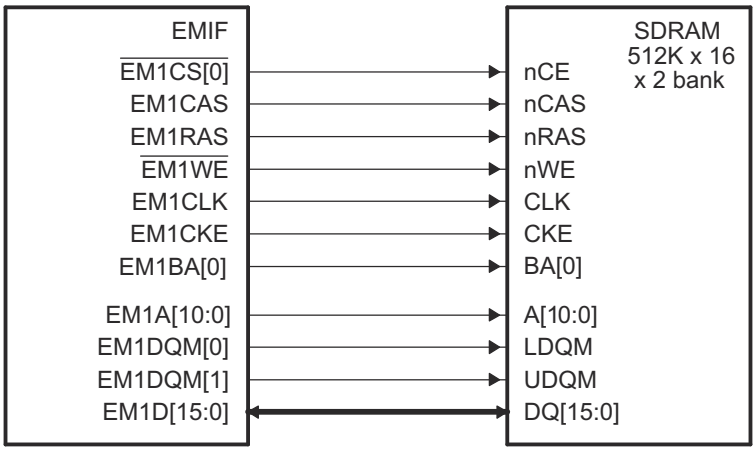



**Figure 25-5. EMIF to 512K × 16 × 2 Bank SDRAM Interface**



Copyright © 2024 Texas Instruments Incorporated

**Table 25-7. 16-bit EMIF Address Pin Connections**

| SDRAM Size | Width | Banks | Device | Address Pins |
|---|---|---|---|---|
| 16M bits | ×16 | 2 | SDRAM | A[10:0] |
| | | | EMIF | EM1A[10:0] |
| 64M bits | ×16 | 4 | SDRAM | A[11:0] |
| | | | EMIF | EM1A[11:0] |
| 128M bits | ×16 | 4 | SDRAM | A[11:0] |
| | | | EMIF | EM1A[11:0] |
| 256M bits | x16 | 4 | SDRAM | A[12:0] |
| | | | EMIF | EM1A[12:0] |
| 512M bits | x16 | 4 | SDRAM | A[12:0] |
| | | | EMIF | EM1A[12:0] |

#### 25.2.5.3 SDRAM Configuration Registers

The operation of the EMIF SDRAM interface is controlled by programming the appropriate configuration registers. This section describes the purpose and function of each configuration register, but Section 25.4 can be referred to for a more detailed description of each register, including the default registers values and bit-field positions. The following tables list the four such configuration registers, along with a description of each of their programmable fields.

**Note**

Writing to any of the fields: NM, CL, IBANK, and PAGESIZE in the SDRAM configuration register (SDRAM_CR) causes the EMIF to abandon whatever the EMIF is currently doing and trigger the SDRAM initialization procedure described in Section 25.2.5.4.



Copyright © 2024 Texas Instruments Incorporated

**Table 25-8. Description of the SDRAM Configuration Register (SDRAM_CR)**

| Parameter | Description |
|---|---|
| SR | This bit controls entering and exiting of the self-refresh mode |
| PD | This bit controls entering and exiting of the power-down mode. If both SR and PD bits are set, the EMIF goes into self-refresh mode. |
| PDWR | Perform refreshes during power down. Writing a 1 to this bit causes the EMIF to exit the power-down state and issue an AUTO REFRESH command every time Refresh May level is set. This bit must be set along with PD when entering power-down mode. |
| NM | **Narrow Mode.** This bit defines the width of the data bus between the EMIF and the attached SDRAM device. When set to 1, the data bus is set to 16-bits. When set to 0, the data bus is set to 32-bits. |
| CL | **CAS latency.** This field defines the number of clock cycles between when an SDRAM issues a READ command and when the first piece of data appears on the bus. The value in this field is sent to the attached SDRAM device using the LOAD MODE REGISTER command during the SDRAM initialization procedure as described in Section 25.2.5.4. Only, values of 2h (CAS latency = 2) and 3h (CAS latency = 3) are supported and must be written to this field. While updating the CL field, BIT11_9LOCK bit field must be set to 1 simultaneously. |
| IBANK | **Number of Internal SDRAM Banks.** This field defines the number of banks inside the attached SDRAM devices in the following way:<br>• When IBANK = 0, 1 internal bank is used<br>• When IBANK = 1h, 2 internal banks are used<br>• When IBANK = 2h, 4 internal banks are used<br>This field value affects the mapping of logical addresses to the SDRAM row, column, and bank addresses. See Section 25.2.5.11 for details. |
| PAGESIZE | **Page Size.** This field defines the internal page size of the attached SDRAM devices in the following way:<br>• When PAGESIZE = 0, 256-word pages are used<br>• When PAGESIZE = 1h, 512-word pages are used<br>• When PAGESIZE = 2h, 1024-word pages are used<br>• When PAGESIZE = 3h, 2048-word pages are used<br>This field value affects the mapping of logical addresses to the SDRAM row, column, and bank addresses. See Section 25.2.5.11 for details. |

**Table 25-9. Description of the SDRAM Refresh Control Register (SDRAM_RCR)**

| Parameter | Description |
|---|---|
| RR | **Refresh Rate**. This field controls the rate at which attached SDRAM devices are refreshed. The following equation can be used to determine the required value of RR for an SDRAM device:<br>• RR = $f_{EM1CLK}$ / (Required SDRAM Refresh Rate)<br>More information about the operation of the SDRAM refresh controller is found in Section 25.2.5.6. |

**Table 25-10. Description of the SDRAM Timing Register (SDRAM_TR)**

| Parameter | Description |
|---|---|
| T_RFC<br>T_RP<br>T_RCD<br>T_WR<br>T_RAS<br>T_RC<br>T_RRD | **SDRAM Timing Parameters.** These fields configure the EMIF to comply with the AC timing requirements of the attached SDRAM devices. This allows the EMIF to avoid violating SDRAM timing constraints and to more efficiently schedule operations. More details about each of these parameters can be found in the SDRAM_TR register description. These parameters must be set to satisfy the corresponding timing requirements found in the SDRAM data sheet. |



Copyright © 2024 Texas Instruments Incorporated

**Table 25-11. Description of the SDRAM Self Refresh Exit Timing Register (SDR_EXT_TMNG)**

| Parameter | Description |
|---|---|
| T_XS | **Self Refresh Exit Parameter.** The T_XS field of this register informs the EMIF about the minimum number of EM1CLK cycles required between exiting self-refresh and issuing any command. This parameter must be set to satisfy the $t_{XSR}$ value for the attached SDRAM device. |

#### 25.2.5.4 SDRAM Auto-Initialization Sequence

The EMIF automatically performs an SDRAM initialization sequence, regardless of whether the EMIF is interfaced to an SDRAM device, when either of the following two events occur:

- The EMIF comes out of reset. No memory accesses to the SDRAM and asynchronous interfaces are performed until this auto-initialization is complete.
- A write is performed to any of the three least-significant bytes of the SDRAM configuration register (SDRAM_CR)

An SDRAM initialization sequence consists of the following steps:

1. If the initialization sequence is activated by a write to SDRAM_CR, and if any of the SDRAM banks are open, the EMIF issues a PRE command with EM1A[10] held high to indicate all banks. This is done so that the maximum ACTV to PRE timing for an SDRAM is not violated.
2. The EMIF drives EM1SDCKE high and begins continuously issuing NOP commands until eight SDRAM refresh intervals have elapsed. An SDRAM refresh interval is equal to the value of the RR field of the SDRAM refresh control register (SDRAM_RCR), divided by the frequency of EM1CLK (RR/$f_{EM1CLK}$). This step is used to avoid violating the power-up constraint of most SDRAM devices that requires 200 µs (sometimes 100 µs) between receiving stable Vdd and CLK and the issuing of a PRE command. Depending on the frequency of EM1CLK, this step could be insufficient to avoid violating the SDRAM constraint. See Section 25.2.5.5 for more information.
3. After the refresh intervals have elapsed, the EMIF issues a PRE command with EM1A[10] held high to indicate all banks.
4. The EMIF issues eight AUTO REFRESH commands.
5. The EMIF issues the LMR command with the EM1A[9:0] pins set as described in Table 25-12.
6. Finally, the EMIF performs a refresh cycle, which consists of the following steps:
   a. Issuing a PRE command with EM1A[10] held high if any banks are open
   b. Issuing an REF command

**Table 25-12. SDRAM LOAD MODE REGISTER Command**

| EM1A[9:7] | EM1A[6:4] | EM1A[3] | EM1A[2:0] |
|---|---|---|---|
| 0 (Write bursts are of the programmed burst length in EM1A[2:0]) | These bits control the CAS latency of the SDRAM and are set according to CL field in the SDRAM configuration register (SDRAM_CR) as follows:<br>• If CL = 2, EM1A[6:4] = 2h (CAS latency = 2)<br>• If CL = 3, EM1A[6:4] = 3h (CAS latency = 3) | 0 (Sequential Burst Type. Interleaved Burst Type not supported) | These bits control the burst length of the SDRAM and are set according to the NM field in the SDRAM configuration register (SDRAM_CR) as follows:<br>• If NM = 0, EM1A[2:0] = 2h (Burst Length = 4)<br>• If NM = 1, EM1A[2:0] = 3h (Burst Length = 8) |



Copyright © 2024 Texas Instruments Incorporated

#### 25.2.5.5 SDRAM Configuration Procedure

There are two different SDRAM configuration procedures. Although the EMIF automatically performs the SDRAM initialization sequence described in Section 25.2.5.4 when coming out of reset, follow one of the procedures before performing any EMIF memory requests.

Procedure A must be followed if the SDRAM power-up constraint was not violated during the SDRAM auto-initialization sequence detailed in Section 25.2.5.4 on coming out of Reset. The SDRAM power-up constraint specifies that 200µs (sometimes 100µs) must exist between receiving stable Vdd and CLK and the issuing of a PRE command.

Procedure B must be followed if the SDRAM power-up constraint was violated. The 200µs (100µs) SDRAM power-up constraint is violated, if the frequency of EM1CLK is greater than 50MHz (100MHz for 100µs SDRAM power-up constraint) during SDRAM Auto-Initialization Sequence. Procedure B must be followed if there is any doubt that the power-up constraint was not met.

**Procedure A** — Following is the procedure to be followed if the SDRAM power-up constraint was not violated:

1. Place the SDRAM into self-refresh mode by setting the SR bit of SDRAM_CR to 1. The SDRAM can be placed into self-refresh mode when changing the frequency of the EM1CLK to avoid incurring the 200µs power-up constraint again.
2. Configure the desired EMIF1 clock (EM1CLK) frequency. The frequency of the memory clock must meet the timing requirements in the SDRAM manufacturer's documentation and the timing limitations shown in the electrical specifications of the device data sheet.
3. Remove the SDRAM from self-refresh mode by clearing the SR bit of the SDRAM_CR to 0.
4. Program SDRAM_TR and SDR_EXT_TMNG to satisfy the timing requirements for the attached SDRAM device. The timing parameters must be taken from the SDRAM data sheet.
5. Program the RR field of SDRAM_RCR to match that of the attached device's refresh interval. See Section 25.2.5.6.1 details on determining the appropriate value.
6. Program the SDRAM_CR to match the characteristics of the attached SDRAM device. This causes the auto-initialization sequence in Section 25.2.5.4 to be re-run. This second initialization generally takes much less time due to the increased frequency of EM1CLK.

**Procedure B** — Following is the procedure to be followed if the SDRAM power-up constraint was violated:

1. Configure the desired EM1CLK clock frequency. The frequency of the memory clock must meet the timing requirements in the SDRAM manufacturer's documentation and the timing limitations shown in the electrical specifications of the device data sheet.
2. Program SDRAM_TR and SDR_EXT_TMNG to satisfy the timing requirements for the attached SDRAM device. The timing parameters must be taken from the SDRAM data sheet.
3. Program the RR field of the SDRAM_RCR such that the following equation is satisfied: $(RR \times 8)/(f_{EM1CLK})$ > 200µs (sometimes 100µs). For example, an EM1CLK frequency of 100MHz requires setting RR to 2501 (9C5h) or higher to meet a 200µs constraint.
4. Program the SDRAM_CR to match the characteristics of the attached SDRAM device. This causes the auto-initialization sequence in Section 25.2.5.4 to be re-run with the new value of RR.
5. Perform a read from the SDRAM to make sure that step 5 of this procedure occurs after the initialization process has completed. Alternatively, wait for 200µs instead of performing a read.
6. Finally, program the RR field to match that of the attached device's refresh interval. See Section 25.2.5.6.1 details on determining the appropriate value.

After following the above procedure, the EMIF is ready to perform accesses to the attached SDRAM device.



Copyright © 2024 Texas Instruments Incorporated

#### 25.2.5.6 EMIF Refresh Controller

An SDRAM device requires that each of the rows be refreshed at a minimum required rate. The EMIF can meet this constraint by performing auto refresh cycles at or above this required rate. An auto-refresh cycle consists of issuing a PRE command to all banks of the SDRAM device followed by issuing a REFR command. To inform the EMIF of the required rate for performing auto refresh cycles, the RR field of the SDRAM refresh control register (SDRAM_RCR) must be programmed. The EMIF uses this value along with two internal counters to automatically perform auto refresh cycles at the required rate. The auto-refresh cycles cannot be disabled, even if the EMIF is not interfaced with an SDRAM. The remainder of this section details the EMIF's refresh scheme and provides an example for determining the appropriate value to place in the RR field of the SDRAM_RCR.

The two counters used to perform auto-refresh cycles are a 13-bit refresh interval counter and a 4-bit refresh backlog counter. At reset and upon writing to the RR field, the refresh interval counter is loaded with the value from RR field and begins decrementing, by one, each EMIF clock cycle. When the refresh interval counter reaches zero, the following actions occur:

- The refresh interval counter is reloaded with the value from the RR field and restarts decrementing.
- The 4-bit refresh backlog counter increments unless the counter has already reached the maximum value.

The refresh backlog counter records the number of auto refresh cycles that the EMIF currently has outstanding. This counter is decremented by one each time an auto refresh cycle is performed and incremented by one each time the refresh interval counter expires. The refresh backlog counter saturates at the values of 0000b and 1111b. The EMIF uses the refresh backlog counter to determine the urgency with which an auto refresh cycle must be performed. The four levels of urgency are described in Table 25-13. This refresh scheme allows the required refreshes to be performed with minimal impact on access requests.

**Table 25-13. Refresh Urgency Levels**

| Urgency Level | Refresh Backlog Counter Range | Action Taken |
|---|---|---|
| Refresh May | 1-3 | An auto-refresh cycle is performed only if the EMIF has no requests pending and none of the SDRAM banks are open. |
| Refresh Release | 4-7 | An auto-refresh cycle is performed if the EMIF has no requests pending, regardless of whether any SDRAM banks are open. |
| Refresh Need | 8-11 | An auto-refresh cycle is performed at the completion of the current access unless there are read requests pending. |
| Refresh Must | 12-15 | Multiple auto-refresh cycles are performed at the completion of the current access until the Refresh Release urgency level is reached. At that point, the EMIF can begin servicing any new read or write requests. |



Copyright © 2024 Texas Instruments Incorporated

#### 25.2.5.6.1 Determining the Appropriate Value for the RR Field

The value that must be programmed into the RR field of the SDRAM_RCR must can be calculated by using the frequency of the EM1CLK signal ($f_{EM1CLK}$) and the required refresh rate of the SDRAM ($f_{Refresh}$). The following formula can be used:

$RR = f_{EM1CLK} / f_{Refresh}$

The SDRAM data sheet often communicates the required SDRAM Refresh Rate in terms of the number of REFR commands required in a given time interval. The required SDRAM Refresh Rate in the formula above can therefore be calculated by dividing the number of required cycles per time interval ($n_{cycles}$) by the time interval given in the data sheet ($t_{Refresh\ Period}$) :

$f_{Refresh} = n_{cycles} / t_{Refresh\ Period}$

Combining these formulas, the value that must be programmed into the RR field can be computed as:

$RR = f_{EM1CLK} \times t_{Refresh\ Period} / n_{cycles}$

The following example illustrates calculating the value of RR. Given that:

- $f_{EM1CLK}$ = 100MHz (frequency of EMIF clock)
- $t_{Refresh\ Period}$ = 64ms (required refresh interval of the SDRAM)
- $n_{cycles}$ = 8192 (number of cycles in a refresh interval for the SDRAM)

RR can be calculated as:

RR = 100MHz × 64ms/8192

RR = 781.25

RR = 782 cycles = 30Eh cycles

### 25.2.5.7 Self-Refresh Mode

The EMIF can be programmed to enter the self-refresh state by setting the SR bit of SDRAM_CR to 1. This causes the EMIF to issue the SLFR command after completing any outstanding SDRAM access requests and clearing the refresh backlog counter by performing one or more auto refresh cycles. This places the attached SDRAM device into self-refresh mode in which the EMIF consumes a minimal amount of power while performing the refresh cycles.

While in the self-refresh state, the EMIF continues to service asynchronous bank requests and register accesses as normal, with one caveat. The EMIF does not park the data bus following a read to asynchronous memory while in the self-refresh state. Instead, the EMIF tri-states the data bus. Therefore, it is not recommended to perform asynchronous read operations while the EMIF is in the self-refresh state to prevent floating inputs on the data bus. More information about data bus parking can be found in Section 25.2.7.

The EMIF exits from the self-refresh state, if either of the following events occur:

- The SR bit of SDRAM_CR is cleared to 0.
- An SDRAM accesses is requested.

The EMIF exits from the self-refresh state by driving EM1SDCKE high and performing an auto refresh cycle.

The attached SDRAM device must also be placed into self-refresh mode when changing the frequency of EM1CLK. If the frequency of EM1CLK changes while the SDRAM is not in self-refresh mode, Procedure B in Section 25.2.5.5 must be followed to reinitialize the device.



Copyright © 2024 Texas Instruments Incorporated

#### 25.2.5.8 Power-Down Mode

To support low-power modes, the EMIF can be requested to issue a POWER DOWN command to the SDRAM by setting the PD bit in the SDRAM configuration register (SDRAM_CR). When this bit is set, the EMIF continues normal operation until all outstanding memory access requests have been serviced and the SDRAM refresh backlog (if there is one) has been cleared. At this point, the EMIF enters the power-down state. Upon entering this state, the EMIF issues a POWER DOWN command (same as a NOP command but driving the EM1SDCKE low on the same cycle). The EMIF then maintains the EM1SDCKE low until the EMIF exits the power-down state.

Since the EMIF services the refresh backlog before the EMIF enters the power-down state, all internal banks of the SDRAM are closed (precharged) prior to issuing the POWER DOWN command. Therefore, the EMIF only supports precharge power-down. The EMIF does not support active power-down, where internal banks of the SDRAM are open (active) before the POWER DOWN command is issued.

During the power-down state, the EMIF services the SDRAM, asynchronous memory, and register accesses as normal, returning to the power-down state upon completion.

The PDWR bit in the SDRAM_CR indicates whether the EMIF must perform refreshes in power-down state. If the PDWR bit is set, the EMIF exits the power-down state every time the Refresh Must level is set, performs AUTO REFRESH commands to the SDRAM, and returns back to the power-down state. This evenly distributes the refreshes to the SDRAM in power-down state. If the PDWR bit is not set, the EMIF does not perform any refreshes to the SDRAM. Therefore, the data integrity of the SDRAM is not maintained upon power-down exit, if the PDWR bit is not set.

If the PD bit is cleared while in the power-down state, the EMIF comes out of the power-down state. The EMIF:

- Drives EM1SDCKE high
- Enters the idle state



Copyright © 2024 Texas Instruments Incorporated

#### 25.2.5.9 SDRAM Read Operation

When the EMIF receives a read request to the SDRAM from one of the requesters listed in Section 25.2.2, the EMIF performs one or more read access cycles. A read access cycle begins with the issuing of the ACTV command to select the desired bank and row of the SDRAM device. After the row has been opened, the EMIF proceeds to issue a READ command while specifying the desired bank and column address. EM1A[10] is held low during the READ command to avoid auto-precharging. The READ command signals the SDRAM device to output data from the specified address while EMIF issues NOP commands. Following a READ command, the CL field of the SDRAM configuration register (SDRAM_CR) defines how many delay cycles are present before the read data appears on the data bus. This is referred to as the CAS latency.

Figure 25-6 shows the signal waveforms for a basic SDRAM read operation in which multiple data is read from a single page. On this device, burst accesses are not supported; hence, the EMIF issues a READ command for each data access. Only when the EMIF SDRAM interface is configured to 16-bit by setting the NM bit of the SDRAM configuration register (SDRAM_CR) to 1 and CPU (or any other master) does a 32-bit READ access, a burst access is issued with a size of two.

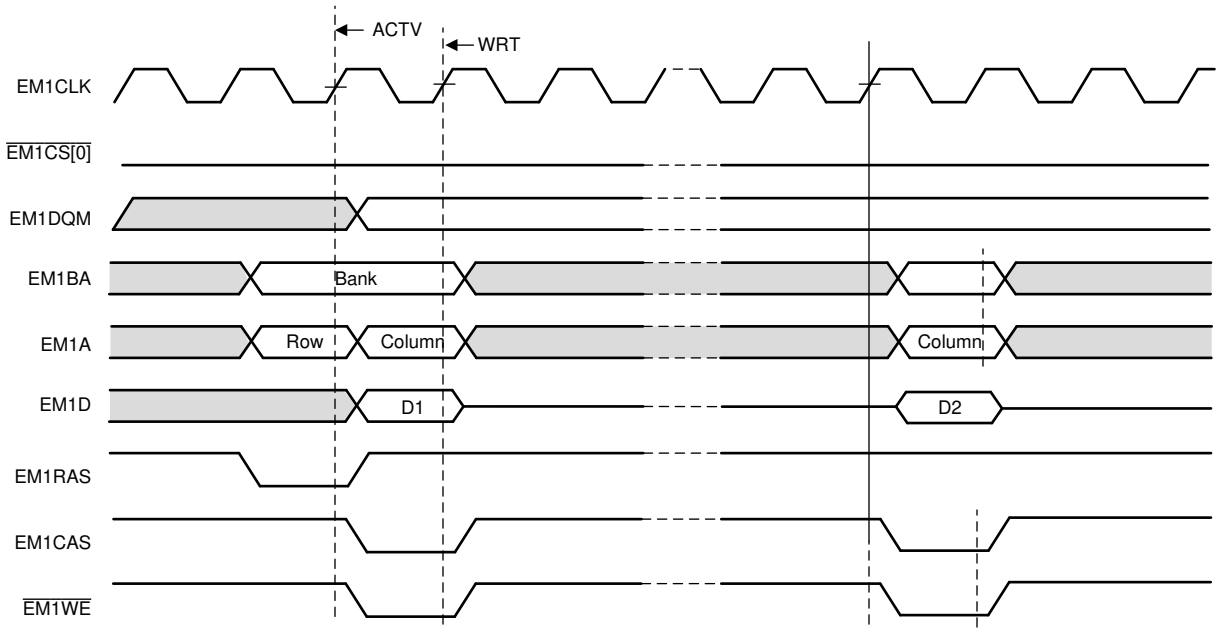



**Figure 25-6. Timing Waveform for Basic SDRAM Read Operation**

Several other pins are also active during a read access. The EM1DQM[x:0] pins are driven low during the READ commands and are kept low during the NOP commands that correspond to the burst request. The state of the other EMIF pins during each command can be found in Table 25-6.

The EMIF schedules the commands based on the timing information that is provided to the EMIF in the SDRAM timing register (SDRAM_TR). The values for the timing parameters in this register must be chosen to satisfy the timing requirements listed in the SDRAM data manual. The EMIF uses this timing information to avoid violating any timing constraints related to issuing commands. This is commonly accomplished by inserting NOP commands between various commands during an access. Refer to the register description of SDRAM_TR in the SDTIMER register for more details on the various timing parameters.



Copyright © 2024 Texas Instruments Incorporated

#### 25.2.5.10 SDRAM Write Operations

When the EMIF receives a write request to SDRAM from one of the requesters listed in Section 25.2.2, the EMIF performs one or more write-access cycles. A write-access cycle begins with the issuing of the ACTV command to select the desired bank and row of the SDRAM device. After the row has been opened, the EMIF proceeds to issue a WRT command while specifying the desired bank and column address. EM1A[10] is held low during the WRT command to avoid auto-precharging. The WRT command signals the SDRAM device to start writing the given data to the specified address while the EMIF issues NOP commands. On this device, burst accesses are not supported; hence, the EMIF issues a WRITE command for each data access. Only when the EMIF SDRAM interface is configured to 16-bit by setting the NM bit of the SDRAM configuration register (SDRAM_CR) to 1 and CPU (or any other master) does a 32bit WRITE access, a burst access is issued with size of two.

Figure 25-7 shows the signal waveforms for a basic SDRAM write operation.

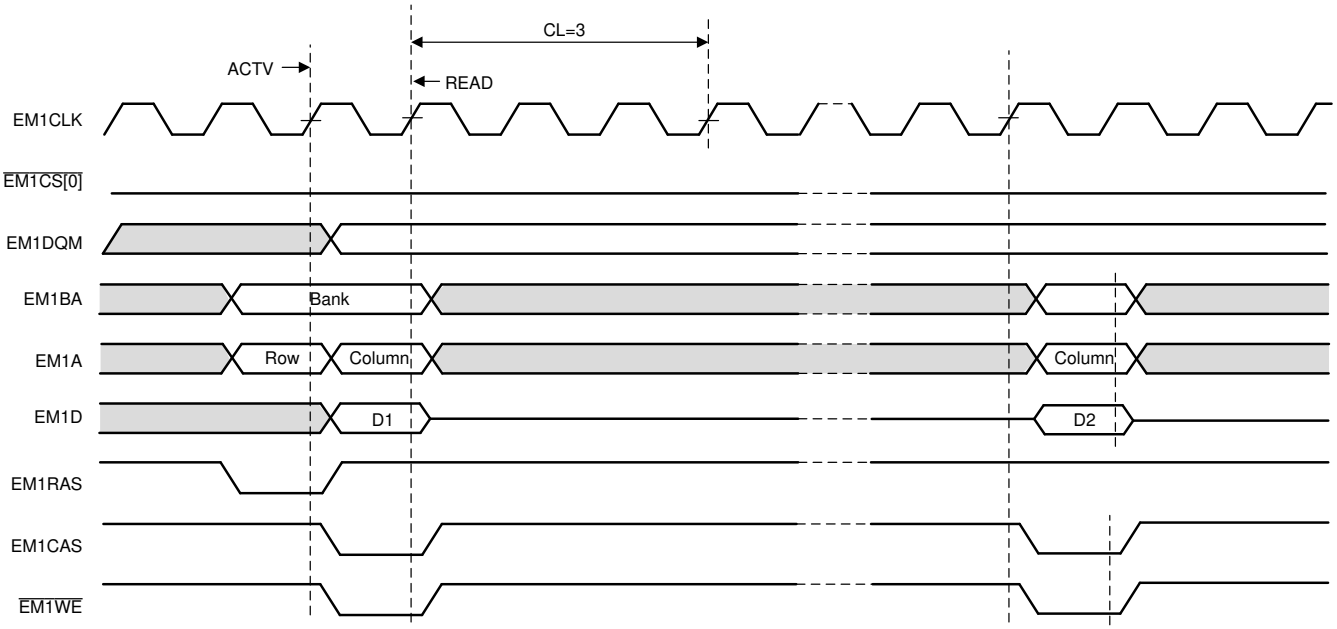



**Figure 25-7. Timing Waveform for Basic SDRAM Write Operation**

The EMIF truncates a series of bursting data if the remaining addresses of the burst are not part of the write request. The EMIF can truncate the burst in three ways:

- By issuing another WRT to the same page
- By issuing a PRE command to prepare for accessing a different page of the same bank
- By issuing a BT command to prepare for accessing a page in a different bank

Several other pins are also active during a write access. The EM1DQM[x:0] pins are driven to select which bytes of the data word are written to the SDRAM device. The pins are also used to mask out entire undesired data words during a burst access. The state of the other EMIF pins during each command can be found in Table 25-6.

The EMIF schedules the commands based on the timing information that is provided to the EMIF in the SDRAM timing register (SDRAM_TR). The values for the timing parameters in this register must be chosen to satisfy the timing requirements listed in the SDRAM data sheet. The EMIF uses this timing information to avoid violating any timing constraints related to issuing commands. This is commonly accomplished by inserting NOP commands during various cycles of an access. Refer to the register description of SDRAM_TR in the SDTIMR register for more details on the various timing parameters.



Copyright © 2024 Texas Instruments Incorporated

### 25.2.5.11 Mapping from Logical Address to EMIF Pins

When the EMIF receives an SDRAM access request, the EMIF must convert the address of the access into the appropriate signals to send to the SDRAM device. The details of this address mapping are shown in Table 25-14 for 32-bit operation and in Table 25-15 for 16-bit operation. Using the settings of the IBANK and PAGESIZE fields of the SDRAM configuration register (SDRAM_CR), the EMIF determines which bits of the logical address are mapped to the SDRAM row, column, and bank addresses.

As the logical address is incremented by one halfword (16-bit operation), the column address is likewise incremented by one until a page boundary is reached. When the logical address increments across a page boundary, the EMIF moves into the same page in the next bank of the attached device by incrementing the bank address EM1BA and resetting the column address. The page in the previous bank is left open until necessary to close the page. This method of traversal through the SDRAM banks helps maximize the number of open banks inside of the SDRAM and results in an efficient use of the device. There is no limitation on the number of banks that can be open at one time, but only one page within a bank can be open at a time.

The EMIF uses the EM1DQM[3:0] pins during a WRT command to mask out selected bytes or entire words. The EM1DQM[3:0] pins are always low during a READ command.

**Table 25-14. Mapping from Logical Address to EMIF Pins for 32-bit SDRAM**

| IBANK | PAGESIZE | Logical Address | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 31:27 | 26 | 25 | 24 | 23 | 22 | 21:14 | 13 | 12 | 11 | 10 | 9 | 8:1 | 0 |
| 0 | 0 | - | | | | | | Row Address | | | | | | Col Address | EM1DQM[2]/EM1DQM[3] |
| 1 | 0 | - | | | | | | Row Address | | | | | EM1BA[0] | Col Address | EM1DQM[2]/EM1DQM[3] |
| 2 | 0 | - | | | | | | Row Address | | | | EM1BA[1:0] | | Col Address | EM1DQM[2]/EM1DQM[3] |
| 0 | 1 | - | | | | | | Row Address | | | | | Column Address | | EM1DQM[2]/EM1DQM[3] |
| 1 | 1 | - | | | | | | Row Address | | | | EM1BA[0] | Column Address | | EM1DQM[2]/EM1DQM[3] |
| 2 | 1 | - | | | | | | Row Address | | | EM1BA[1:0] | | Column Address | | EM1DQM[2]/EM1DQM[3] |
| 0 | 2 | - | | | | | | Row Address | | | | Column Address | | | EM1DQM[2]/EM1DQM[3] |
| 1 | 2 | - | | | | | | Row Address | | | EM1BA[0] | Column Address | | | EM1DQM[2]/EM1DQM[3] |
| 2 | 2 | - | | | | | | Row Address | | EM1BA[1:0] | | Column Address | | | EM1DQM[2]/EM1DQM[3] |
| 0 | 3 | - | | | | | | Row Address | | | Column Address | | | | EM1DQM[2]/EM1DQM[3] |
| 1 | 3 | - | | | | | | Row Address | | EM1BA[0] | Column Address | | | | EM1DQM[2]/EM1DQM[3] |
| 2 | 3 | - | | | | | | Row Address | | EM1BA[1:0] | Column Address | | | | EM1DQM[2]/EM1DQM[3] |

**Table 25-15. Mapping from Logical Address to EMIF Pins for 16-bit SDRAM**

| IBANK | PAGESIZE | Logical Address | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 31:26 | 25 | 24 | 23 | 22 | 21 | 20:13 | 12 | 11 | 10 | 9 | 8 | 7:0 |
| 0 | 0 | - | | | | | | Row Address | | | | | | Col Address |
| 1 | 0 | - | | | | | | Row Address | | | | | EM1BA[0] | Col Address |
| 2 | 0 | - | | | | | | Row Address | | | | EM1BA[1:0] | | Col Address |
| 0 | 1 | - | | | | | | Row Address | | | | | Column Address | |
| 1 | 1 | - | | | | | | Row Address | | | | EM1BA[0] | Column Address | |
| 2 | 1 | - | | | | | | Row Address | | | EM1BA[1:0] | | Column Address | |
| 0 | 2 | - | | | | | | Row Address | | | | Column Address | | |
| 1 | 2 | - | | | | | | Row Address | | | EM1BA[0] | Column Address | | |
| 2 | 2 | - | | | | | | Row Address | | EM1BA[1:0] | | Column Address | | |
| 0 | 3 | - | | | | | | Row Address | | | Column Address | | | |
| 1 | 3 | - | | | | | | Row Address | | EM1BA[0] | Column Address | | | |
| 2 | 3 | - | | | | | | Row Address | | EM1BA[1:0] | Column Address | | | |



Copyright © 2024 Texas Instruments Incorporated

**Note**

The upper bit of the row address is used only when addressing 256-Mbit and 512-Mbit SDRAM memories.

### 25.2.6 Asynchronous Controller and Interface

The EMIF easily interfaces to a variety of asynchronous devices including NOR Flash and SRAM. It can be operated in two major modes (see Table 25-16):

- Normal Mode
- Select Strobe Mode

**Table 25-16. Normal Mode vs. Select Strobe Mode**

| Mode | Function of EM1DQM pins | Operation of $\overline{\text{EM1CS}}$[4:2] |
|---|---|---|
| Normal Mode | Byte enables | Active during the entire asynchronous access cycle |
| Select Strobe Mode | Byte enables | Active only during the strobe period of an access cycle |

The first mode of operation is normal mode, in which the EM1DQM pins of the EMIF function as byte enables. In this mode, the $\overline{\text{EM1CS}}$[4:2] pins behave as typical chip select signals, remaining active for the duration of the asynchronous access. See Section 25.2.6.1 for an example interface with multiple 8-bit devices.

The second mode of operation is select strobe mode, in which the $\overline{\text{EM1CS}}$[4:2] pins act as a strobe, active only during the strobe period of an access. In this mode, the EM1DQM pins of the EMIF function as standard byte enables for reads and writes. A summary of the differences between the two modes of operation are shown in Table 25-16. Refer to Section 25.2.6.4 for the details of asynchronous operations in normal mode, and to Section 25.2.6.5 for the details of asynchronous operations in select strobe mode. The EMIF hardware defaults to normal mode, but can be manually switched to select strobe mode by setting the SS bit in the asynchronous *m* (m = 1, 2, 3, or 4) configuration register (CE*n*CFG) (*n* = 2, 3, or 4). Throughout the chapter, *m* can hold the values 1, 2, 3 or 4; and *n* can hold the values 2, 3, or 4.

In both normal mode and select strobe mode, EMIF can be configured to operate in a sub-mode called NAND Flash Mode. In NAND Flash Mode, the EMIF is able to calculate an error correction code (ECC) for transfers up to 512 bytes.

The EMIF also provides configurable cycle timing parameters and an extended wait mode that allows the connected device to extend the strobe period of an access cycle. The following sections describe the features related to interfacing with external asynchronous devices.

#### 25.2.6.1 Interfacing to Asynchronous Memory

Figure 25-8 shows the EMIF's external pins used in interfacing with an asynchronous device. In $\overline{\text{EM1CS}}$[n], n = 2, 3, or 4.

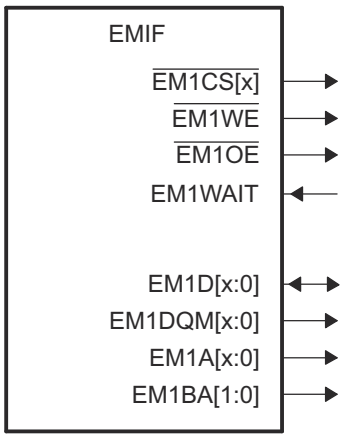


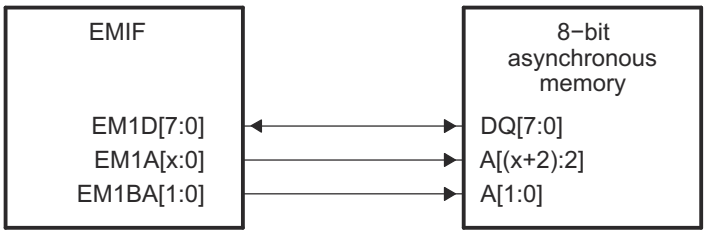

**Figure 25-8. EMIF Asynchronous Interface**



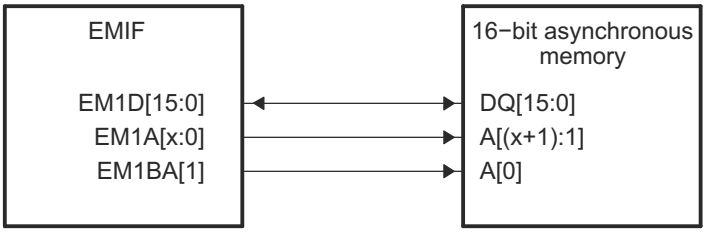
Copyright © 2024 Texas Instruments Incorporated

Of special note is the connection between the EMIF and the external device's address bus. The EMIF address pin EM1A[0] always provides the least-significant bit of a 32-bit word address. Therefore, when interfacing to a 16-bit or 8-bit asynchronous device, the EM1BA[1] and EM1BA[0] pins provide the least-significant bits of the halfword or byte address, respectively. Figure 25-9 and Figure 25-10 show the mapping between the EMIF and the connected device's data and address pins for various programmed data bus widths. The data bus width can be configured in the asynchronous *n* configuration register (ASYNC_CS*n*_CR).

Figure 25-10 shows an interface between the EMIF and an external memory with byte enables. The EMIF must be operated in either normal mode or select strobe mode when using this interface, so that the EM1DQM signals operate as byte enables.




a) EMIF to 8-bit memory interface




b) EMIF to 16-bit memory interface

**Figure 25-9. EMIF to 8-bit/16-bit Memory Interface**

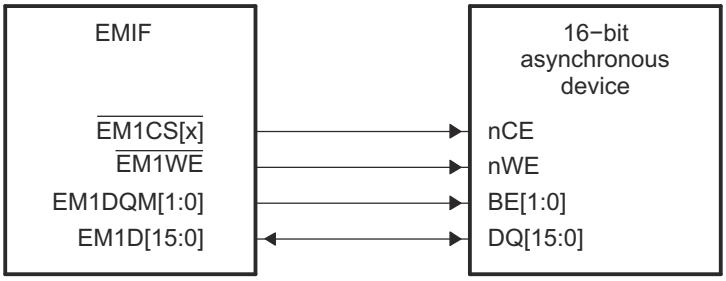



**Figure 25-10. Common Asynchronous Interface**



Copyright © 2024 Texas Instruments Incorporated

#### 25.2.6.2 Accessing Larger Asynchronous Memories

If a device such as a large asynchronous Flash needs to be attached to the EMIF, then GPIO pins can be used to control the Flash device upper address lines.

#### 25.2.6.3 Configuring EMIF for Asynchronous Accesses

The operation of the EMIF's asynchronous interface can be configured by programming the appropriate register fields. The reset value and bit position for each register field can be found in Section 25.4. The following tables list the register fields that can be programmed and describe the purpose of each field. These registers can be programmed prior to accessing the external memory, and the transfer following a write to these registers uses the new configuration.

**Table 25-17. Description of the Asynchronous *m* Configuration Register (ASYNC_CSn_CR)**

| Parameter | Description |
|---|---|
| SS | **Select Strobe mode.** This bit selects the EMIF's mode of operation in the following way:<br>• SS = 0 selects Normal Mode<br>– EM1DQM pins function as byte enables<br>– $\overline{\text{EM1CS}}$[4:2] active for duration of access<br>• SS = 1 selects Select Strobe Mode<br>– EM1DQM pins function as byte enables<br>– $\overline{\text{EM1CS}}$[4:2] acts as a strobe. |
| EW | **Extended Wait Mode enable.**<br>• EW = 0 disables Extended Wait Mode<br>• EW = 1 enables Extended Wait Mode<br>When set to 1, the EMIF enables the Extended Wait Mode in which the strobe width of an access cycle can be extended in response to the assertion of the EM1WAIT pin. The WP*n* bit in the asynchronous wait cycle configuration register (AWCC) controls the polarity of the EM1WAIT pin. Extended Wait Mode must not be used while in NAND Flash Mode. See Section 25.2.6.6 for more details on this mode of operation. |
| W_SETUP/R_SETUP | **Read/Write setup widths.**<br>These fields define the number of EMIF clock cycles of setup time for the address pins (EM1A), byte enables (EM1DQM), and asynchronous chip enable ($\overline{\text{EM1CS}}$[4:2]) before the read strobe pin (EM103) or write strobe pin ($\overline{\text{EM1WE}}$) falls, minus one cycle. For writes, the W_SETUP field also defines the setup time for the data pins (EM1D). Refer to the asynchronous device 's data sheet to determine the appropriate setting for this field. |
| W_STROBE/R_STROBE | **Read/Write strobe widths.**<br>These fields define the number of EMIF clock cycles between the falling and rising of the read strobe pin (EM103) or write strobe pin (EM1WEn), minus one cycle. If Extended Wait Mode is enabled by setting the EW field in the asynchronous *n* configuration register (ASYNC_CS*n*_CR), these fields must be set to a value greater than zero. Refer to the data manual of the external asynchronous device to determine the appropriate setting for this field. |
| W_HOLD/R_HOLD | **Read/Write hold widths.**<br>These fields define the number of EMIF clock cycles of hold time for the address pins (EM1A and EM1BA), byte enables (EM1DQM), and asynchronous chip enable ($\overline{\text{EM1CS}}$[4:2]) after the read strobe pin (EM103) or write strobe pin ($\overline{\text{EM1WE}}$) rises, minus one cycle. For writes, the W_HOLD field also defines the hold time for the data pins (EM1D). Refer to the data manual of the external asynchronous device to determine the appropriate setting for this field. |
| TA | **Minimum turnaround time.**<br>This field defines the minimum number of EMIF clock cycles between asynchronous reads and writes, minus one cycle. The purpose of this feature is to avoid contention on the bus. The value written to this field also determines the number of cycles that are inserted between asynchronous accesses and SDRAM accesses. Refer to the data manual of the external asynchronous device to determine the appropriate setting for this field. |



Copyright © 2024 Texas Instruments Incorporated

**Table 25-17. Description of the Asynchronous *m* Configuration Register (ASYNC_CSn_CR) (continued)**

| Parameter | Description |
|---|---|
| ASIZE | **Asynchronous Device Bus Width.**<br>This field determines the data bus width of the asynchronous interface in the following way:<br>• ASIZE = 0 selects an 8-bit bus<br>• ASIZE = 1 selects a 16-bit bus<br>• ASIZE = 2 selects a 32-bit bus<br>The configuration of ASIZE determines the function of the EM1A and EM1BA pins as described in Section 25.2.6.1. This field also determines the number of external accesses required to fulfill a request generated by one of the sources mentioned in Section 25.2.2. For example, a request for a 32-bit word requires four external access when ASIZE = 0. Refer to the data manual of the external asynchronous device to determine the appropriate setting for this field. |

**Table 25-18. Description of the Asynchronous Wait Cycle Configuration Register (ASYNC_WCCR)**

| Parameter | Description |
|---|---|
| WP*n* | **EM_WAIT Polarity.**<br>• WP*n* = 0 selects active-low polarity<br>• WP*n* = 1 selects active-high polarity<br>When set to 1, the EMIF waits if the EM1WAIT pin is high. When cleared to 0, the EMIF waits if the EM1WAIT pin is low. The EMIF must have the Extended Wait Mode enabled for the EM1WAIT pin to affect the width of the strobe period. The polarity of the EM1WAIT signal is not programmable in NAND Flash Mode. |
| MAX_EXT_WAIT | **Maximum Extended Wait Cycles.**<br>This field configures the number of EMIF clock cycles the EMIF waits for the EM1WAIT pin to be deactivated during the strobe period of an access cycle. The maximum number of EMIF clock cycles the EMIF waits is determined by the following formula:<br>Maximum Extended Wait Cycles = (MAX_EXT_WAIT + 1) × 16<br>If the EM1WAIT pin is not deactivated within the time specified by this field, the EMIF resumes the access cycle, registering whatever data is on the bus and proceeding to the hold period of the access cycle. This situation is referred to as an Asynchronous Timeout. An Asynchronous Timeout generates an interrupt, if the interrupt has been enabled in the EMIF interrupt mask set register (INT_MSK_SET). Refer to Section 25.2.9.1 for more information about EMIF interrupts.<br>Extended Wait Mode can not be used while in NAND Flash Mode. |

**Table 25-19. Description of EMIF Interrupt Mask Set Register (INT_MSK_SET)**

| Parameter | Description |
|---|---|
| WR_MASK_SET | **Wait Rise Mask Set.**<br>Writing a 1 enables an interrupt to be generated when a rising edge on EM1WAIT occurs. |
| AT_MASK_SET | **Asynchronous Timeout Mask Set.**<br>Writing a 1 to this bit enables an interrupt to be generated when an Asynchronous Timeout occurs. |



Copyright © 2024 Texas Instruments Incorporated

**Table 25-20. Description of EMIF Interrupt Mast Clear Register (INT_MSK_CLR)**

| Parameter | Description |
|---|---|
| WR_MASK_CLR | **Wait Rise Mask Clear.**<br>Writing a 1 to this bit disables the interrupt, clearing the WR_MASK_SET bit in EMIF interrupt mask set register (INT_MSK_SET). |
| AT_MASK_CLR | **Asynchronous Timeout Mask Clear.**<br>Writing a 1 to this bit prevents an interrupt from being generated when an Asynchronous Timeout occurs. |

**Note**

The EMIF performs SDRAM refreshes even if the SDRAM interface is not used. If using only the ASRAM interface, then SDRAM refreshes impact the ASRAM performance. To avoid this, set PD=1 in the SDRAM_CR register (Emif1Regs.SDRAM_CR.bit.PD=1). This bit can be updated only if there are no pending EMIF accesses.

#### 25.2.6.4 Read and Write Operations in Normal Mode

Normal mode is the asynchronous interface's default mode of operation. It is selected when the SS bit in the asynchronous *n* configuration register (ASYNC_CS*n*_CR) is cleared to 0. In this mode, the EM1DQM pins operate as byte enables. Section 25.2.6.4.1 and Section 25.2.6.4.2 explain the details of read and write operations while in normal mode.

##### *25.2.6.4.1 Asynchronous Read Operations (Normal Mode)*

**Note**

During an entire asynchronous read operation, the $\overline{\text{EM1WE}}$ pin is driven high.

An asynchronous read is performed when any of the requesters mentioned in Section 25.2.2 request a read from the attached asynchronous memory. After the request is received, a read operation is initiated once the request becomes the EMIF's highest priority task, according to the priority scheme detailed in Section 25.2.13. In the event that the read request cannot be serviced by a single access cycle to the external device, multiple access cycles are performed by EMIF until the entire request is fulfilled. The details of an asynchronous read operation in normal mode are described in Table 25-21. Also, Figure 25-11 shows an example timing diagram of a basic read operation.

**Table 25-21. Asynchronous Read Operation in Normal Mode**

| Time Interval | Pin Activity in Normal Mode |
|---|---|
| Turnaround period | Once the read operation becomes the highest priority task for EMIF, the EMIF waits for the programmed number of turn-around cycles before proceeding to the setup period of the operation. The number of wait cycles is taken directly from the TA field of the asynchronous *n* configuration register (ASYNC_CS*n*_CR). Between each access (write or read), the EMIF inserts two cycles of delay even though TA field is programmed as 0.<br>After the EMIF has waited for the turnaround cycles to complete, the EMIF again checks to make sure that the read operation is still the highest priority task. If so, the EMIF proceeds to the setup period of the operation. If the read operation is no longer the highest priority task, the EMIF terminates the operation. |
| Start of the setup period | The following actions occur at the start of the setup period:<br>• The setup, strobe, and hold values are set according to the R_SETUP, R_STROBE, and R_HOLD values in ASYNC_CS*n*_CR.<br>• The address pins EM1A and EM1BA become valid and carry the values described in Section 25.2.6.1.<br>• $\overline{\text{EM1CS}}$[4:2] falls to enable the external device (if not already low from a previous operation) |



Copyright © 2024 Texas Instruments Incorporated

**Table 25-21. Asynchronous Read Operation in Normal Mode (continued)**

| Time Interval | Pin Activity in Normal Mode |
|---|---|
| Strobe period | The following actions occur during the strobe period of a read operation:<br>1. $\overline{\text{EM1OE}}$ falls at the start of the strobe period<br>2. On the rising edge of the clock that is concurrent with the end of the strobe period:<br>• $\overline{\text{EM1OE}}$ rises<br>• The data on the EM1Dx bus is sampled by EMIF.<br><br>In Figure 25-11, EM1WAIT is inactive. If EM1WAIT is instead activated, the strobe period can be extended by the external device to give it more time to provide the data. Section 25.2.6.6 contains more details on using the EM1WAIT pin. |
| End of the hold period | At the end of the hold period:<br>• The address pins EM1A and EM1BA become invalid<br>• $\overline{\text{EM1CS}}$[4:2] rises (if no more operations are required to complete the current request)<br><br>The EMIF can be required to issue additional read operations to a device with a small data bus width to complete an entire word access. In this case, the EMIF immediately re-enters the setup period to begin another operation without incurring the turn-round cycle delay. The setup, strobe, and hold values are not updated in this case. If the entire word access has been completed, the EMIF returns to the previous state unless another asynchronous request has been submitted and is currently the highest priority task. If this is the case, EMIF instead enters directly into the turnaround period for the pending read or write operation. |

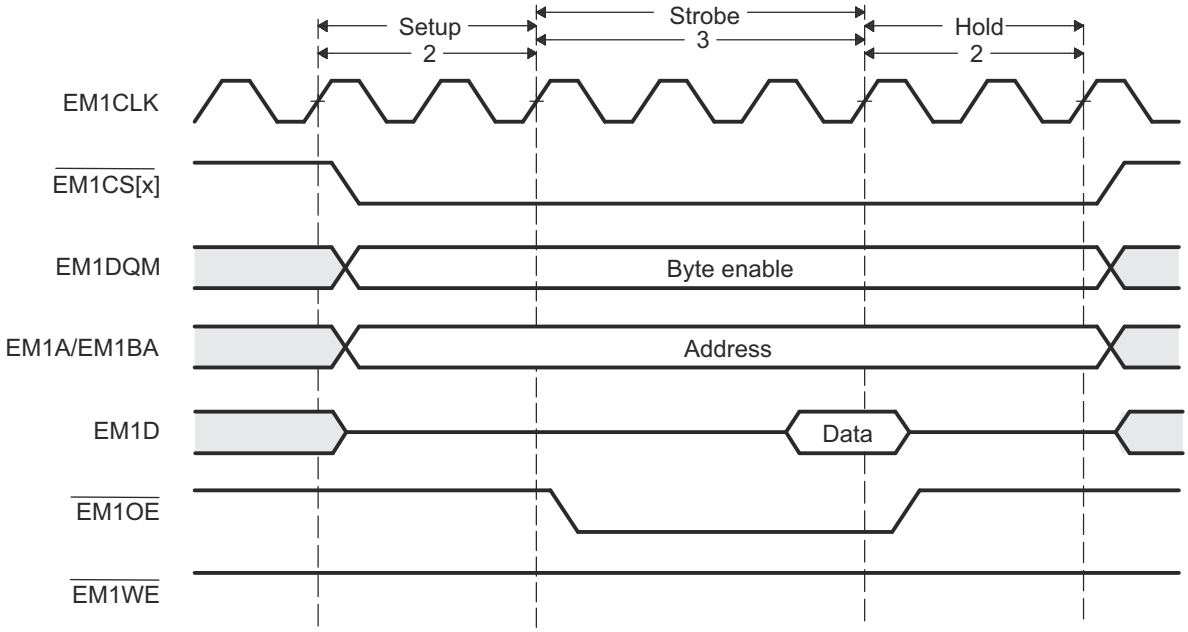



**Figure 25-11. Timing Waveform of an Asynchronous Read Cycle in Normal Mode**



Copyright © 2024 Texas Instruments Incorporated

#### 25.2.6.4.2 Asynchronous Write Operations (Normal Mode)

**Note**

During an entire asynchronous write operation, the EM1OE pin is driven high.

An asynchronous write is performed when any of the requesters mentioned in Section 25.2.2 request a write to memory in the asynchronous bank of EMIF. After the request is received, a write operation is initiated once the request becomes the EMIF's highest priority task, according to the priority scheme detailed in Section 25.2.13. In the event that the write request cannot be serviced by a single access cycle to the external device, multiple access cycles are performed by the EMIF until the entire request is fulfilled. The details of an asynchronous write operation in normal mode are described in Table 25-22. Also, Figure 25-12 shows an example timing diagram of a basic write operation.

**Table 25-22. Asynchronous Write Operation in Normal Mode**

| Time Interval | Pin Activity in Normal Mode |
|---|---|
| Turnaround period | Once the write operation becomes the highest priority task for the EMIF, the EMIF waits for the programmed number of turn-around cycles before proceeding to the setup period of the operation. The number of wait cycles is taken directly from the TA field of the asynchronous *n* configuration register (ASYNC_CS*n*_CR). Between each access (write or read) EMIF inserts two cycles of delay even though TA field is programmed as 0.<br>After the EMIF has waited for the turn-around cycles to complete, the EMIF again checks to make sure that the write operation is still the highest priority task. If so, the EMIF proceeds to the setup period of the operation. If the write operation is no longer the highest priority task, EMIF terminates the operation. |
| Start of the setup period | The following actions occur at the start of the setup period:<br>• The setup, strobe, and hold values are set according to the W_SETUP, W_STROBE, and W_HOLD values in ASYNC_CS*n*_CR.<br>• The address pins EM1A and EM1BA and the data pins EM1Dx become valid. The EM1A and EM1BA pins carry the values described in Section 25.2.6.1.<br>• EM1CS[4:2] falls to enable the external device (if not already low from a previous operation). |
| Strobe period | The following actions occur at the start of the strobe period of a write operation:<br>1. EM1WE falls<br>2. The EM1DQM pins become valid as byte enables.<br><br>The following actions occur on the rising edge of the clock that is concurrent with the end of the strobe period:<br>1. EM1WE rises<br>2. The EM1DQM pins deactivate<br><br>In Figure 25-12, EM1WAIT is inactive. If EM1WAIT is instead activated, the strobe period can be extended by the external device to give it more time to accept the data. Section 25.2.6.6 contains more details on using the EM1WAIT pin. |
| End of the hold period | At the end of the hold period:<br>• The address pins EM1Ax and EM1BAx become invalid<br>• The data pins become invalid<br>• EM1CS[n] (n = 2, 3, or 4) rises (if no more operations are required to complete the current request)<br><br>The EMIF can be required to issue additional write operations to a device with a small data bus width to complete an entire word access. In this case, the EMIF immediately re-enters the setup period to begin another operation without incurring the turnaround cycle delay. The setup, strobe, and hold values are not updated in this case. If the entire word access has been completed, the EMIF returns to the previous state unless another asynchronous request has been submitted and is currently the highest priority task. If this is the case, the EMIF instead enters directly into the turnaround period for the pending read or write operation. |



Copyright © 2024 Texas Instruments Incorporated

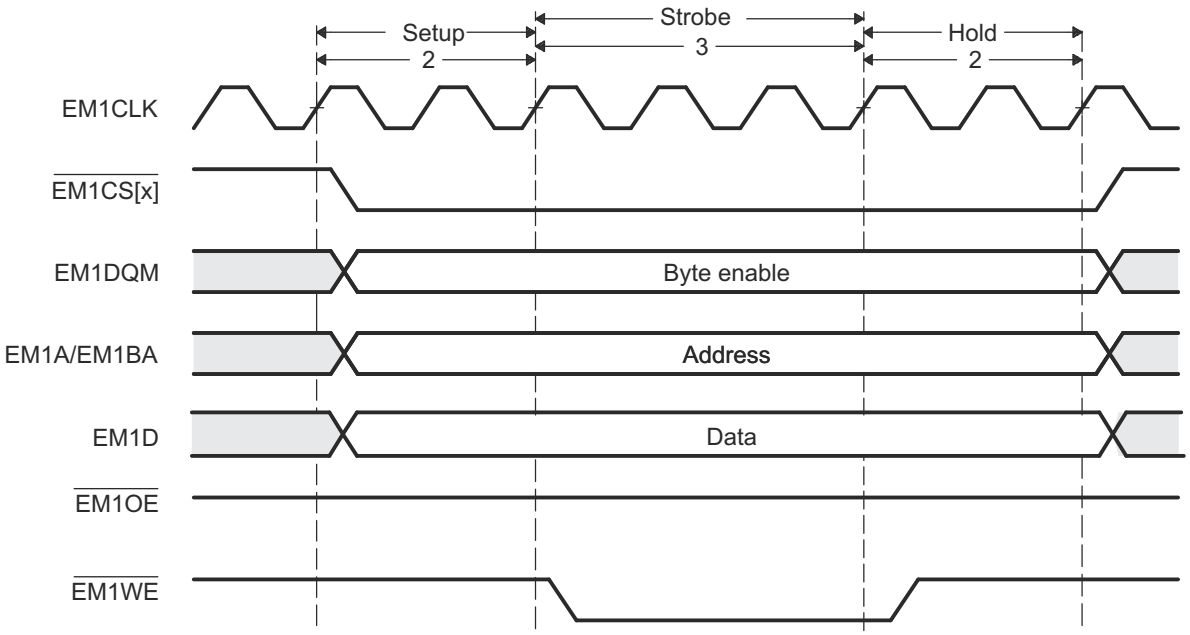



**Figure 25-12. Timing Waveform of an Asynchronous Write Cycle in Normal Mode**



Copyright © 2024 Texas Instruments Incorporated

#### 25.2.6.5 Read and Write Operation in Select Strobe Mode

Select Strobe Mode is the EMIF's second mode of operation. It is selected when the SS bit of the asynchronous *n* configuration register (ASYNC_CSn_CR) is set to 1. In this mode, the EM1DQM pins operate as byte enables and the $\overline{\text{EM1CS}}$[n] (n = 2, 3, or 4) pin is only active during the strobe period of an access cycle. Section 25.2.6.4.1 and Section 25.2.6.4.2 explain the details of read and write operations while in select strobe mode.

##### *25.2.6.5.1 Asynchronous Read Operations (Select Strobe Mode)*

**Note**

During the entirety of an asynchronous read operation, the EM1WEn pin is driven high.

An asynchronous read is performed when any of the requesters mentioned in Section 25.2.2 request a read from the attached asynchronous memory. After the request is received, a read operation is initiated once the request becomes the EMIF's highest priority task, according to the priority scheme detailed in Section 25.2.13. In the event that the read request cannot be serviced by a single access cycle to the external device, multiple access cycles are performed by the EMIF until the entire request is fulfilled. The details of an asynchronous read operation in select strobe mode are described in Table 25-23. Also, Figure 25-13 shows an example timing diagram of a basic read operation.

**Table 25-23. Asynchronous Read Operation in Select Strobe Mode**

| Time Interval | Pin Activity in Select Strobe Mode |
|---|---|
| Turnaround period | Once the read operation becomes the highest priority task for the EMIF, the EMIF waits for the programmed number of turnaround cycles before proceeding to the setup period of the operation. The number of wait cycles is taken directly from the TA field of the asynchronous *n* configuration register (ASYNC_CSn_CR). Between each access (Write or Read) EMIF inserts two cycles of delay even though TA field is programmed as 0.<br>After the EMIF has waited for the turn-around cycles to complete, the EMIF again checks to make sure that the read operation is still the highest priority task. If so, the EMIF proceeds to the setup period of the operation. If the read operation is no longer the highest priority task, the EMIF terminates the operation. |
| Start of the setup period | The following actions occur at the start of the setup period:<br>• The setup, strobe, and hold values are set according to the R_SETUP, R_STROBE, and R_HOLD values in ASYNC_CS*n*_CR.<br>• The address pins EM1A and EM1BA become valid and carry the values described in Section 25.2.6.1.<br>• The EM1DQM pins become valid as byte enables. |
| Strobe period | The following actions occur during the strobe period of a read operation:<br>1. $\overline{\text{EM1CS}}$[n] (n = 2, 3, or 4) and $\overline{\text{EM1OE}}$ fall at the start of the strobe period<br>2. On the rising edge of the clock that is concurrent with the end of the strobe period:<br>  • $\overline{\text{EM1CS}}$[n] (n = 2, 3, or 4) and $\overline{\text{EM10E}}$ rise<br>  • The data on the EM1D bus is sampled by EMIF.<br>In Figure 25-13, EM1WAIT is inactive. If EM1WAIT is instead activated, the strobe period can be extended by the external device to give more time to provide the data. Section 25.2.6.6 contains more details on using the EM1WAIT pin. |
| End of the hold period | At the end of the hold period:<br>• The address pins EM1A and EM1BA become invalid<br>• The EM1DQM pins become invalid<br>The EMIF can be required to issue additional read operations to a device with a small data bus width to complete an entire word access. In this case, the EMIF immediately re-enters the setup period to begin another operation without incurring the turnaround cycle delay. The setup, strobe, and hold values are not updated in this case. If the entire word access has been completed, the EMIF returns to the previous state unless another asynchronous request has been submitted and is currently the highest priority task. If this is the case, the EMIF instead enters directly into the turnaround period for the pending read or write operation. |



Copyright © 2024 Texas Instruments Incorporated

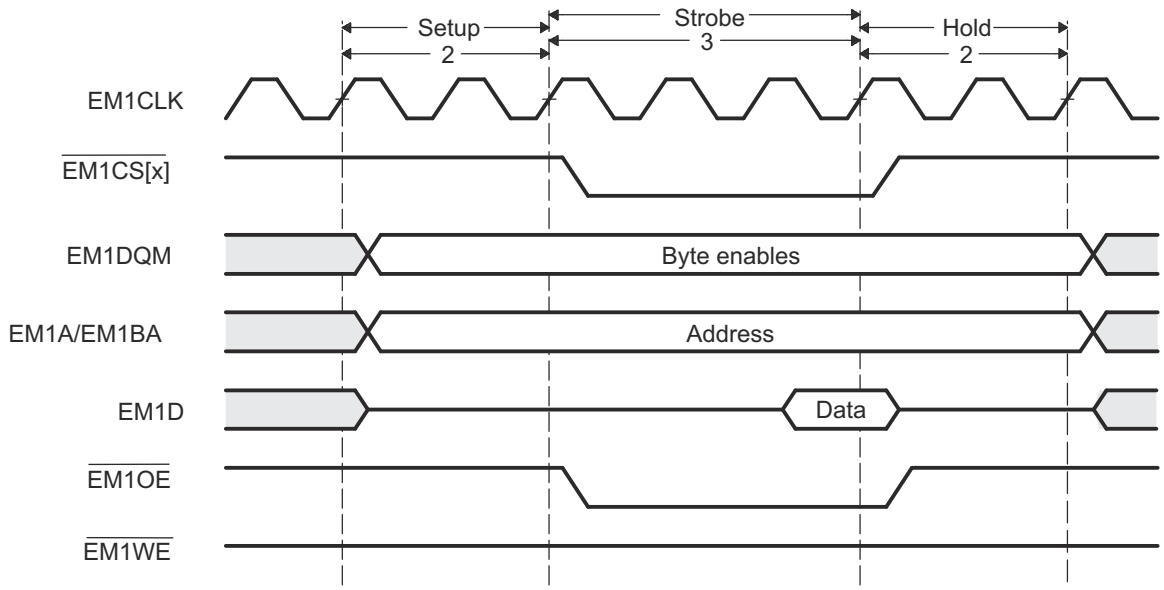



**Figure 25-13. Timing Waveform of an Asynchronous Read Cycle in Select Strobe Mode**



Copyright © 2024 Texas Instruments Incorporated

#### 25.2.6.5.2 Asynchronous Write Operations (Select Strobe Mode)

**Note**

During the entirety of an asynchronous write operation, the $\overline{\text{EM1OE}}$ pin is driven high.

An asynchronous write is performed when any of the requesters mentioned in Section 25.2.2 request a write to memory in the asynchronous bank of EMIF. After the request is received, a write operation is initiated once the request becomes the EMIF's highest priority task, according to the priority scheme detailed in Section 25.2.13. In the event that the write request cannot be serviced by a single access cycle to the external device, multiple access cycles are performed by the EMIF until the entire request is fulfilled. The details of an asynchronous write operation in select strobe mode are described in Table 25-24. Also, Figure 25-14 shows an example timing diagram of a basic write operation.

**Table 25-24. Asynchronous Write Operation in Select Strobe Mode**

| Time Interval | Pin Activity in Select Strobe Mode |
|---|---|
| Turnaround period | Once the write operation becomes the highest priority task for the EMIF, the EMIF waits for the programmed number of turnaround cycles before proceeding to the setup period of the operation. The number of wait cycles is taken directly from the TA field of the asynchronous *n* configuration register (ASYNC_CSn_CR). Between each access (Write or Read) EMIF inserts two cycles of delay even though TA field is programmed as 0.<br>After the EMIF has waited for the turnaround cycles to complete, the EMIF again checks to make sure that the write operation is still the highest priority task. If so, the EMIF proceeds to the setup period of the operation. If the write operation is no longer the highest priority task, the EMIF terminates the operation. |
| Start of the setup period | The following actions occur at the start of the setup period:<br>• The setup, strobe, and hold values are set according to the W_SETUP, W_STROBE, and W_HOLD values in ASYNC_CSn_CR.<br>• The address pins EM1A and EM1BA and the data pins EM1D become valid. The EM1A and EM1BA pins carry the values described in Section 25.2.6.1.<br>• The EM1DQM pins become active as byte enables. |
| Strobe period | The following actions occur at the start of the strobe period of a write operation:<br>• $\overline{\text{EM1CS}}$[n] (n = 2, 3, or 4) and $\overline{\text{EM1WE}}$ fall<br>The following actions occur on the rising edge of the clock which is concurrent with the end of the strobe period:<br>• $\overline{\text{EM1CS}}$[n] (n = 2, 3, or 4) and $\overline{\text{EM1WE}}$ rise<br>In Figure 25-14, EM1WAIT is inactive. If EM1WAIT is instead activated, the strobe period can be extended by the external device to give more time to accept the data. Section 25.2.6.6 contains more details on using the EM1WAIT pin. |
| End of the hold period | At the end of the hold period:<br>• The address pins EM1A and EM1BA become invalid<br>• The data pins become invalid<br>• The EM1DQM pins become invalid<br>The EMIF can be required to issue additional write operations to a device with a small data bus width to complete an entire word access. In this case, the EMIF immediately re-enters the setup period to begin another operation without incurring the turnaround cycle delay. The setup, strobe, and hold values are not updated in this case. If the entire word access has been completed, the EMIF returns to the previous state unless another asynchronous request has been submitted and is currently the highest priority task. If this is the case, the EMIF instead enters directly into the turnaround period for the pending read or write operation. |



Copyright © 2024 Texas Instruments Incorporated

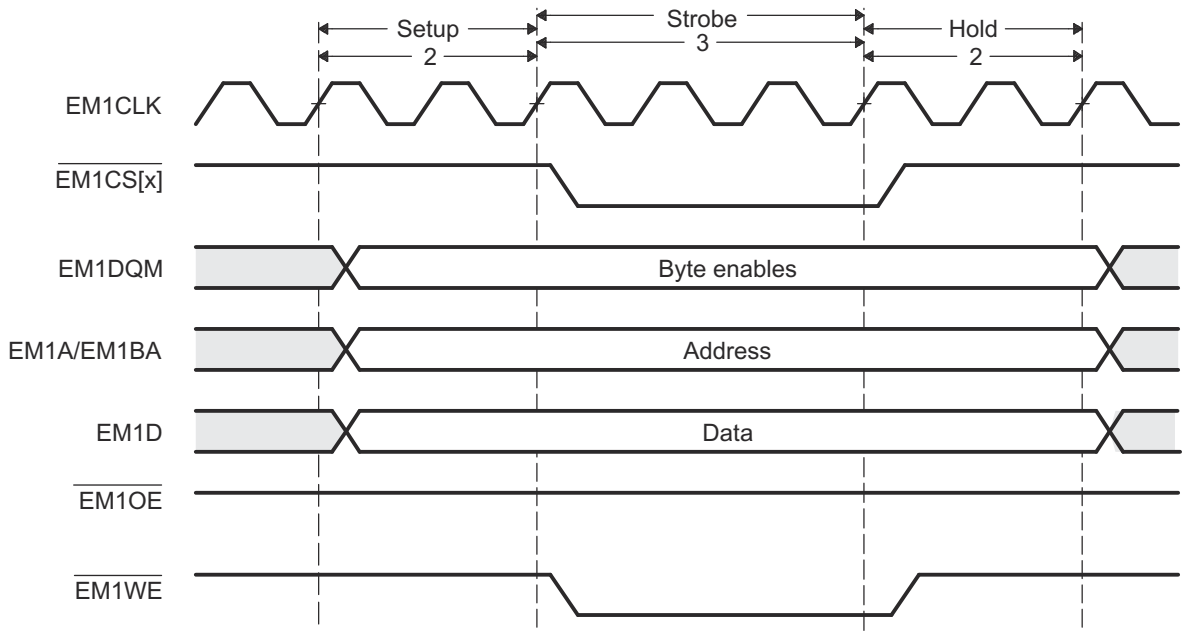



**Figure 25-14. Timing Waveform of an Asynchronous Write Cycle in Select Strobe Mode**

#### 25.2.6.6 Extended Wait Mode and the EM1WAIT Pin

The EMIF supports the extended wait mode. This is a mode that the external asynchronous device can assert control over the length of the strobe period. The extended wait mode can be entered by setting the EW bit in the asynchronous *n* configuration register (ASYNC_CS*n*_CR (*n* = 2, 3, or 4). When this bit is set, the EMIF monitors the EM1WAIT pin to determine if the attached device wishes to extend the strobe period of the current access cycle beyond the programmed number of clock cycles.

When the EMIF detects that the EM1WAIT pin has been asserted, the EMIF begins inserting extra strobe cycles into the operation until the EM1WAIT pin is deactivated by the external device. The EMIF then returns to the last cycle of the programmed strobe period and the operation proceeds as usual from this point. Refer to the device data sheet for details on the timing requirements of the EM1WAIT signal.

The EM1WAIT pin cannot be used to extend the strobe period indefinitely. The programmable MAX_EXT_WAIT field in the asynchronous wait cycle configuration register (AWCC) determines the maximum number of EM1CLK cycles the strobe period can be extended beyond the programmed length. When the counter expires, the EMIF proceeds to the hold period of the operation regardless of the state of the EM1WAIT pin. The EMIF can also generate an interrupt upon expiration of this counter. See Section 25.2.9.1 for details on enabling this interrupt.

For the EMIF to function properly in the extended wait mode, the WP*n* bit of AWCC must be programmed to match the polarity of the EM1WAIT pin. In the reset state of 1, the EMIF inserts wait cycles when the EM1WAIT pin is sampled high. When set to 0, the EMIF inserts wait cycles only when EM1WAIT is sampled low. This programmability allows for a glueless connection to larger variety of asynchronous devices.

Finally, a restriction is placed on the strobe period timing parameters when operating in extended wait mode. Specifically, the sum of the W_SETUP and W_STROBE fields must be greater than four, and the sum of the R_SETUP and R_STROBE fields must be greater than four for the EMIF to recognize the EM1WAIT pin has been asserted. The W_SETUP, W_STROBE, R_SETUP, and R_STROBE fields are in ASYNC_CS*n*_CR.



Copyright © 2024 Texas Instruments Incorporated

### 25.2.7 Data Bus Parking

The EMIF always drives the data bus to the previous write data value when the EMIF is idle. This feature is called data bus parking. Only when the EMIF issues a read command to the external memory does the EMIF stop driving the data bus. After the EMIF latches the last read data, the EMIF immediately parks the data bus again.

The one exception to this behavior occurs after performing an asynchronous read operation while the EMIF is in the self-refresh state. In this situation, the read operation is not followed by the EMIF parking the data bus. Instead, the EMIF tri-states the data bus. Therefore, it is not recommended to perform asynchronous read operations while the EMIF is in the self-refresh state, to prevent floating inputs on the data bus. External pull-ups, such as 10-kohm resistors, must be placed on the 16 EMIF data bus pins (that do not have internal pull-ups) if required to perform reads in this situation. The precise resistor value must be chosen so that the worst case combined off-state leakage currents do not cause the voltage levels on the associated pins to drop below the high-level input voltage requirement.

For information about the self-refresh state, see Section 25.2.5.7.

### 25.2.8 Reset and Initialization Considerations

The EMIF memory controller has two active-low reset signals, CHIP_RST_n and MOD_G_RST_n. Both these reset signals are driven by the device system reset signal. This device does not offer the flexibility to reset just the EMIF state machine without also resetting the EMIF controller's memory-mapped registers. As soon as the device system reset is released (driven high), the EMIF memory controller immediately begins its initialization sequence. Command and data stored in the EMIF memory controller FIFOs are lost. Refer to Section 25.2 for more information on conditions that can cause a device system reset to be asserted.

When system reset is released, the EMIF automatically begins running the SDRAM initialization sequence described in Section 25.2.5.4. Even though the initialization procedure is automatic, a special procedure, found in Section 25.2.5.5 must still be followed.

### 25.2.9 Interrupt Support

The EMIF supports a single interrupt to the CPU. Section 25.2.9.1 details the generation and internal masking of EMIF interrupts.

#### 25.2.9.1 Interrupt Events

There are three conditions that can cause the EMIF to generate an interrupt to the CPU. These conditions are:

- A rising edge on the EM1WAIT signal (wait rise interrupt)
- An asynchronous time out
- Usage of unsupported addressing mode (line trap interrupt)

The wait rise interrupt occurs when a rising edge is detected on EM1WAIT signal. This interrupt generation is not affected by the WP*n* bit in the asynchronous wait cycle configuration register (ASYNC_WCCR). The asynchronous time out interrupt condition occurs when the attached asynchronous device fails to deassert the EM1WAIT pin within the number of cycles defined by the MAX_EXT_WAIT bit in AWCC (this happens only in extended wait mode). The EMIF supports only linear incrementing and cache line wrap addressing modes . If an access request for an unsupported addressing mode is received, the EMIF sets the LT bit in the EMIF interrupt raw register (INT_RAW) and treats the request as a linear incrementing request.

Only when the interrupt is enabled by setting the appropriate bit (WR_MASK_SET/AT_MASK_SET/ LT_MASK_SET) in the EMIF interrupt mask set register (INT_MSK_SET) to 1, is the interrupt sent to the CPU. Once enabled, the interrupt can be disabled by writing a 1 to the corresponding bit in the EMIF interrupt mask clear register (INT_MSK_CLR). The bit fields in both the INT_MSK_SET and INT_MSK_CLR can be used to indicate whether the interrupt is enabled. When the interrupt is enabled, the corresponding bit field in both the INT_MSK_SET and INT_MSK_CLR have a value of 1; when the interrupt is disabled, the corresponding bit field has a value of 0.



Copyright © 2024 Texas Instruments Incorporated

The EMIF interrupt raw register (INT_RAW) and the IF interrupt mask register (INT_MSK) indicate the status of each interrupt. The appropriate bit (WR/AT/LT) in INT_RAW is set when the interrupt condition occurs, whether or not the interrupt has been enabled. However, the appropriate bit (WR_MASKED/AT_MASKED/LT_MASKED) in INT_MSK is set only when the interrupt condition occurs and the interrupt is enabled. Writing a 1 to the bit in INT_RAW clears the INT_RAW bit as well as the corresponding bit in INT_MSK. Table 25-25 contains a brief summary of the interrupt status and control bit fields. See for complete details on the register fields.

**Table 25-25. Interrupt Monitor and Control Bit Fields**

| Register Name | Bit Name | Description |
|---|---|---|
| EMIF interrupt raw register (INT_RAW) | WR | This bit is set when an rising edge on the EM1WAIT signal occurs. Writing a 1 clears the WR bit as well as the WR_MASKED bit in INT_MSK. |
| | AT | This bit is set when an asynchronous timeout occurs. Writing a 1 clears the AT bit as well as the AT_MASKED bit in INT_MSK. |
| | LT | This bit is set when an unsupported addressing mode is used. Writing a 1 clears LT bit as well as the LT_MASKED bit in INT_MSK. |
| EMIF interrupt mask register (INT_MSK) | WR_MASKED | This bit is set only when a rising edge on the EM1WAIT signal occurs and the interrupt has been enabled by writing a 1 to the WR_MASK_SET bit in INT_MSK_SET. |
| | AT_MASKED | This bit is set only when an asynchronous timeout occurs and the interrupt has been enabled by writing a 1 to the AT_MASK_SET bit in INT_MSK_SET. |
| | LT_MASKED | This bit is set only when line trap interrupt occurs and the interrupt has been enabled by writing a 1 to the LT_MASK_SET bit in INT_MSK_SET. |
| EMIF interrupt mask set register (INT_MSK_SET) | WR_MASK_SET | Writing a 1 to this bit enables the wait rise interrupt. |
| | AT_MASK_SET | Writing a 1 to this bit enables the asynchronous timeout interrupt. |
| | LT_MASK_SET | Writing a 1 to this bit enables the line trap interrupt. |
| EMIF interrupt mask clear register (INT_MSK_CLR) | WR_MASK_CLR | Writing a 1 to this bit disables the wait rise interrupt. |
| | AT_MASK_CLR | Writing a 1 to this bit disables the asynchronous timeout interrupt. |
| | LT_MASK_CLR | Writing a 1 to this bit disables the line trap interrupt. |

### *25.2.10 DMA Event Support*

The EMIF memory controller is a DMA slave peripheral and therefore does not generate DMA events. Data read and write requests can be made directly, by masters and the DMA.

### *25.2.11 EMIF Signal Multiplexing*

For details on the EMIF signal multiplexing, see the *GPIO* chapter, I/O Multiplexing Module section, of this technical reference manual.

### *25.2.12 Memory Map*

For information describing the device memory-map, see the data sheet.



Copyright © 2024 Texas Instruments Incorporated

### 25.2.13 Priority and Arbitration

Section 25.2.2 describes the external prioritization and arbitration among requests from different sources within the microcontroller. The result of this external arbitration is that only one request is presented to the EMIF at a time. Once the EMIF completes a request, the external arbiter then provides the EMIF with the next pending request.

Internally, the EMIF undertakes memory device transactions according to a strict priority scheme. The highest priority events are:

- A device reset.
- A write to any of the three least significant bytes of the SDRAM configuration register (SDRAM_CR).

Either of these events causes the EMIF to immediately commence the initialization sequence as described in Section 25.2.5.4.

Once the EMIF has completed its initialization sequence, the EMIF performs memory transactions according to the following priority scheme (highest priority listed first):

1. If the EMIF's backlog refresh counter is at the Refresh Must urgency level, the EMIF performs multiple SDRAM auto-refresh cycles until the Refresh Release urgency level is reached.
2. If an SDRAM or asynchronous read has been requested, the EMIF performs a read operation.
3. If the EMIF's backlog refresh counter is at the Refresh Need urgency level, the EMIF performs an SDRAM auto-refresh cycle.
4. If an SDRAM or asynchronous write has been requested, the EMIF performs a write operation.
5. If the EMIF's backlog refresh counter is at the Refresh May or Refresh Release urgency level, the EMIF performs an SDRAM auto-refresh cycle.
6. If the value of the SR bit in SDRAM_CR has been set to 1, the EMIF enters the self-refresh state as described in Section 25.2.5.7.

After taking one of the actions listed above, the EMIF then returns to the top of the priority list to determine the next action.

Because the EMIF does not issue auto-refresh cycles when in the self-refresh state, the above priority scheme does not apply when in this state. See Section 25.2.5.7 for details on the operation of the EMIF when in the self-refresh state.

### 25.2.14 System Considerations

This section describes various system considerations to keep in mind when operating the EMIF.

#### 25.2.14.1 Asynchronous Request Times

In a system that interfaces to both SDRAM and asynchronous memory, the asynchronous requests must not take longer than the smaller of the following two values:

- $t_{RAS}$ (typically 120 µs) - to avoid violating the maximum time allowed between issuing an ACTV and PRE command to the SDRAM.
- $t_{Refresh\ Rate}$ × 11 (typically 15.7 µs × 11 = 172.7 µs) - to avoid refresh violations on the SDRAM.

The length of an asynchronous request is controlled by multiple factors, the primary factor being the number of access cycles required to complete the request. For example, an asynchronous request for 4 bytes requires four access cycles using an 8-bit data bus and only two access cycle using a 16-bit data bus. The maximum request size that the EMIF can be sent is 16 words; therefore, the maximum number of access cycles per memory request is 64 when the EMIF is configured with an 8-bit data bus. The length of the individual access cycles that make up the asynchronous request is determined by the programmed setup, strobe, hold, and turnaround values, but can also be extended with the assertion of the EM1WAIT input signal up to a programmed maximum limit. The user must make sure that an entire asynchronous request does not exceed the timing values listed above when also interfacing to an SDRAM device. This can be done by limiting the asynchronous timing parameters.



Copyright © 2024 Texas Instruments Incorporated

### 25.2.15 Power Management

Power dissipation from the EMIF memory controller can be managed by following methods:

- Self-refresh mode
- Power-down mode
- Gating input clocks to the module off

Gating input clocks off to the EMIF memory controller achieves higher power savings when compared to the power savings of self-refresh or power down mode. The input clock to EMIF can be turned off through the use of the Global Clock Module (GCM). Before gating clocks off, the EMIF memory controller must place the SDR SDRAM memory in self-refresh mode. If the external memory requires a continuous clock, the VCLK3 clock domain must not be turned off because this can result in data corruption. See the following subsections for the proper procedures to follow when stopping EMIF memory controller clocks.

#### 25.2.15.1 Power Management Using Self-Refresh Mode

The EMIF memory controller can be placed into a self-refresh state in order to place the attached SDRAM devices into self-refresh mode, which consumes less power for most SDRAM devices. In this state, the attached SDRAM device uses an internal clock to perform its own auto refresh cycles. This maintains the validity of the data in the SDRAM without the need for any external commands. Refer to Section 25.2.5.7 for more details on placing the EMIF into the self-refresh state.

#### 25.2.15.2 Power Management Using Power Down Mode

In the power down mode, the EMIF drives EM1SDCKE low to lower the power consumption. EM1SDCKE goes high when there is a need to send refresh (REFR) commands, after which EM1SDCKE is again driven low. EM1SDCKE remains low until any request arrives. Refer to Section 25.2.5.8 for more details on placing the EMIF in power-down mode.

### 25.2.16 Emulation Considerations

The EMIF remains fully functional during emulation halts in order to allow emulation access to external memory.

## 25.3 Example Configuration

This section presents an example of interfacing the EMIF1 to both an SDR SDRAM device and an asynchronous Flash device.

### 25.3.1 Hardware Interface

Figure 25-15 shows the hardware interface between the EMIF, a Samsung K4S641632H-TC(L)70 64Mb SDRAM device, and a SHARP LH28F800BJE-PTTL90 8Mb Flash memory. The connection between EMIF and the SDRAM is straightforward, but the connection between the EMIF and the Flash deserves a detailed look.

The address inputs for the Flash are provided by three sources. The A[18:0] address inputs are provided by a combination of the EM1A and EM1BA pins according to Section 25.2.6.1, and a set of GPIO pins. The RD/nBY signal from Flash is connected to EM1WAIT pin of the EMIF.

Finally, this example configuration connects the $\overline{\text{EM1WE}}$ pin to the nWE input of the Flash and operates the EMIF in select strobe mode.



Copyright © 2024 Texas Instruments Incorporated

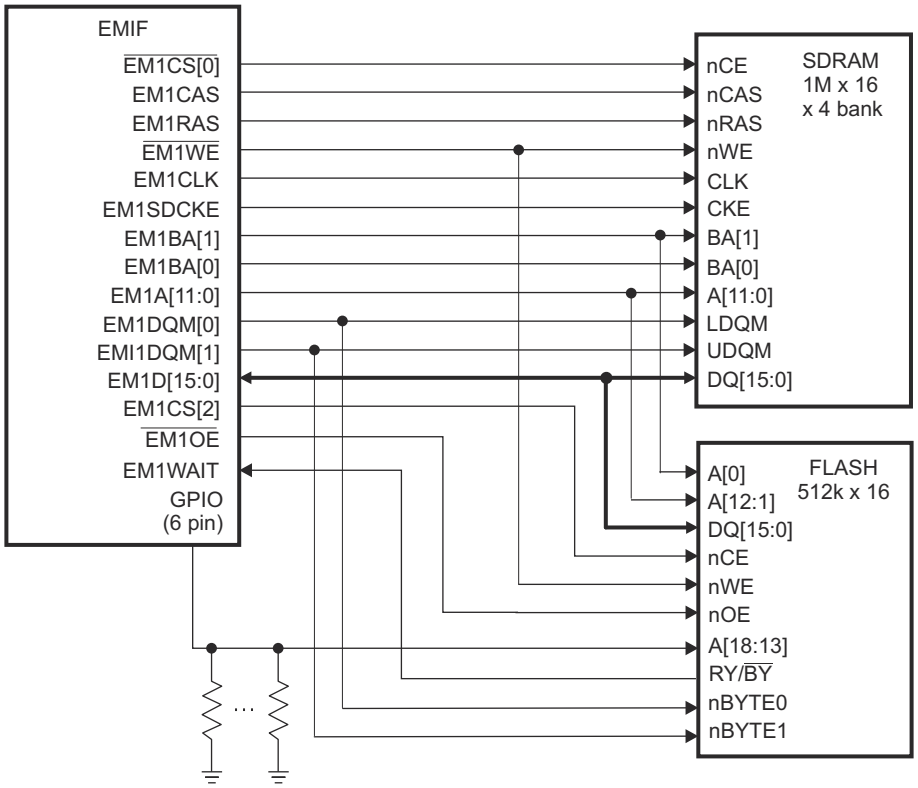



**Figure 25-15. Example Configuration Interface**

### *25.3.2 Software Configuration*

The following sections describe how to configure the EMIF registers and bit fields to interface the EMIF with the Samsung K4S641632H-TC(L)70 SDRAM and the SHARP LH28F800BJE-PTTL90 8Mb Flash memory.

#### 25.3.2.1 Configuring the SDRAM Interface

This section describes how to configure the EMIF to interface with the Samsung K4S641632H-TC(L)70 SDRAM with a clock frequency of $f_{EM1CLK}$ = 100MHz. Procedure A described in Section 25.2.5.5 is followed that assumes that the SDRAM power-up timing constraints were met during the SDRAM auto-initialization sequence after reset.

##### *25.3.2.1.1 PLL Programming for EMIF to K4S641632H-TC(L)70 Interface*

If the system PLL is programmed to provide a SYSCLK frequency > 100MHz, then configure the EMIF1CLKDIV field in the PERSYSCLKDIVSEL register to make EM1CLK = SYSCLK/2 (default configuration). Before doing this, the SDRAM must be placed in self-refresh mode by setting the SR bit in the SDRAM configuration register. Once the EM1CLK frequency has been configured, remove the SDRAM from self-refresh by clearing the SR bit in SDRAM_CR.

**Table 25-26. SR Field Value For EMIF to K4S641632H-TC(L)70 Interface**

| Field | Value | Purpose |
|---|---|---|
| SR | 1 then 0 | To place the EMIF into the self-refresh state |



Copyright © 2024 Texas Instruments Incorporated

#### 25.3.2.1.2 SDRAM Timing Register (SDRAM_TR) Settings for EMIF to K4S641632H-TC(L)70 Interface

The fields of the SDRAM timing register (SDRAM_TR) must be programmed first as described in Table 25-27 to satisfy the required timing parameters for the K4S641632H-TC(L)70. Based on these calculations, a value of 6111 4610h must be written to the SDRAM_TR. Figure 25-16 shows a graphical description of how the SDRAM_TR must be programmed.

**Table 25-27. SDRAM_TR Field Calculations for EMIF to K4S641632H-TC(L)70 Interface**

| Field Name | Formula | Value from K4S641632H-TC(L)70 Data Sheet | Value Calculated for Field |
|---|---|---|---|
| T_RFC | T_RFC >= ($t_{RFC}$ × $f_{EM1CLK}$) - 1 | $t_{RC}$ = 68 ns (min)(1) | 6 |
| T_RP | T_RP >= ($t_{RP}$ × $f_{EM1CLK}$) - 1 | $t_{RP}$ = 20 ns (min) | 1 |
| T_RCD | T_RCD >= ($t_{RCD}$ × $f_{EM1CLK}$) - 1 | $t_{RCD}$ = 20 ns (min) | 1 |
| T_WR | T_WR >= ($t_{WR}$ × $f_{EM1CLK}$) - 1 | $t_{RDL}$ = 2 CLK = 20 ns (min)(2) | 1 |
| T_RAS | T_RAS >= ($t_{RAS}$ × $f_{EM1CLK}$) - 1 | $t_{RAS}$ = 49 ns (min) | 4 |
| T_RC | T_RC >= ($t_{RC}$ × $f_{EM1CLK}$) - 1 | $t_{RC}$ = 68 ns (min) | 6 |
| T_RRD | T_RRD >= ($t_{RRD}$ × $f_{EM1CLK}$) - 1 | $t_{RRD}$ = 14 ns (min) | 1 |

(1) The Samsung data sheet does not specify a $t_{RFC}$ value. Instead, Samsung specifies $t_{RC}$ as the minimum auto refresh period.
(2) The Samsung data sheet does not specify a $t_{WR}$ value. Instead, Samsung specifies $t_{RDL}$ as last data in to row precharge minimum delay.

**Figure 25-16. SDRAM Timing Register (SDRAM_TR)**

| 31 | 30 | 29 | 28 | 27 | 26 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 0110 | | | | | 001 | | 0 | 001 | | | 0 | 001 | | |
| T_RFC | | | | | T_RP | | Rsvd | T_RCD | | | Rsvd | T_WR | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0100 | | | | 0110 | | | | 0 | 001 | | | 0000 | | | |
| T_RAS | | | | T_RC | | | | Rsvd | T_RRD | | | Reserved | | | |



Copyright © 2024 Texas Instruments Incorporated

#### 25.3.2.1.3 SDRAM Self Refresh Exit Timing Register (SDR_EXT_TMNG) Settings for EMIF to K4S641632H-TC(L)70 Interface

The SDRAM self-refresh exit timing register (SDSRETR) must be programmed second to satisfy the $t_{XSR}$ timing requirement from the K4S641632H-TC(L)70 data sheet. Table 25-28 shows the calculation of the proper value to program into the T_XS field of this register. Based on this calculation, a value of 6h must be written to the SDSRETR. Figure 25-17 shows how the SDSRETR must be programmed.

**Table 25-28. RR Calculation for EMIF to K4S641632H-TC(L)70 Interface**

| Field Name | Formula | Value from K4S641632H-TC(L)70 Data Sheet | Value Calculated for Field |
|---|---|---|---|
| T_XS | T_XS >= ($t_{XSR}$ × $f_{EM1CLK}$) - 1 | $t_{RC}$ = 68 ns (min)(1) | 6 |

(1) The Samsung data sheet does not specify a $t_{XSR}$ value. Instead, Samsung specifies $t_{RC}$ as the minimum required time after CKE going high to complete self-refresh exit.

**Figure 25-17. SDRAM Self Refresh Exit Timing Register (SDR_EXT_TMNG)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0000 0000 0000 0000 | | | | | | | | | | | | | | | |
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 000 0000 0000 | | | | | | | | | | | | | 0 0110 | | |
| Reserved | | | | | | | | | | | | | T_XS | | |

#### 25.3.2.1.4 SDRAM Refresh Control Register (SDRAM_RCR) Settings for EMIF to K4S641632H-TC(L)70 Interface

The SDRAM refresh control register (SDRAM_RCR) can next be programmed to satisfy the required refresh rate of the K4S641632H-TC(L)70. Table 25-29 shows the calculation of the proper value to program into the RR field of this register. Based on this calculation, a value of 61Ah must be written to the SDRAM_RCR. Figure 25-18 shows how the SDRAM_RCR must be programmed.

**Table 25-29. RR Calculation for EMIF to K4S641632H-TC(L)70 Interface**

| Field Name | Formula | Values | Value Calculated for Field |
|---|---|---|---|
| RR | RR ≤ $f_{EM1CLK}$ × $t_{Refresh\ Period}$ / $n_{cycles}$ | From SDRAM data sheet: $t_{Refresh\ Period}$ = 64ms; $n_{cycles}$ = 4096 EMIF clock rate: $f_{EM1CLK}$ = 100MHz | RR = 1562 cycles = 61Ah cycles |

**Figure 25-18. SDRAM Refresh Control Register (SDRAM_RCR)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 0000 0000 0000 | | | | | | | | | | | | | | 000 | |
| Reserved | | | | | | | | | | | | | | Reserved | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 000 | | | 0 0110 0001 1010 (61Ah) | | | | | | | | | | | | |
| Reserved | | | RR | | | | | | | | | | | | |



Copyright © 2024 Texas Instruments Incorporated

#### 25.3.2.1.5 SDRAM Configuration Register (SDRAM_CR) Settings for EMIF to K4S641632H-TC(L)70 Interface

Finally, the fields of the SDRAM configuration register (SDRAM_CR) must be programmed as described in Table 25-30 to properly interface with the K4S641632H-TC(L)70 device. Based on these settings, a value of 4720h must be written to the SDRAM_CR. Figure 25-19 shows how the SDRAM_CR must be programmed. The EMIF is now ready to perform read and write accesses to the SDRAM.

**Table 25-30. SDRAM_CR Field Values For EMIF to K4S641632H-TC(L)70 Interface**

| Field | Value | Purpose |
|---|---|---|
| SR | 0 | To avoid placing the EMIF into the self-refresh state |
| NM | 1 | To configure the EMIF for a 16-bit data bus |
| CL | 011b | To select a CAS latency of 3 |
| BIT11_9LOCK | 1 | To allow the CL field to be written |
| IBANK | 010b | To select 4 internal SDRAM banks |
| PAGESIZE | 0 | To select a page size of 256 words |

**Figure 25-19. SDRAM Configuration Register (SDRAM_CR)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 0000 | | | | |
| SR | Reserved | Reserved | Reserved | | | | |

| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|
| 00 0000 | | | | | | 0 | 0 |
| Reserved | | | | | | Reserved | Reserved |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 011 | | | 1 |
| Reserved | NM | Reserved | Reserved | CL | | | BIT11_9LOCK |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 0 | 010 | | | 0 | 000 | | |
| Reserved | IBANK | | | Reserved | PAGESIZE | | |



Copyright © 2024 Texas Instruments Incorporated

#### 25.3.2.2 Configuring the Flash Interface

This section describes how to configure the EMIF to interface with the SHARP LH28F800BJE-PTTL90 8Mb Flash memory with a clock frequency of $f_{EM1CLK}$ = 100MHz.

##### *25.3.2.2.1 Asynchronous 1 Configuration Register (ASYNC_CS2_CFG) Settings for EMIF to LH28F800BJE-PTTL90 Interface*

The asynchronous 1 configuration register (ASYNC_CS2_CFG) is the only register that is necessary to program for this asynchronous interface. The SS bit must be set to 1 to enable select strobe command and the ASIZE field can be set to 1 to select a 16-bit interface. The other fields in this register control the shaping of the EMIF signals, and the proper values can be determined by referring to the AC Characteristics in the Flash data sheet and the device data sheet. Table 25-31 and Table 25-32 show the pertinent AC Characteristics for reads and writes to the Flash device, and Figure 25-20 and Figure 25-21 show the associated timing waveforms. Finally, Figure 25-22 shows programming the ASYNC_CS2_CFG with the calculated values.

**Table 25-31. AC Characteristics for a Read Access**

| AC Characteristic | Device | Definition | Minimum | Maximum | Unit |
|---|---|---|---|---|---|
| $t_{SU}$ | EMIF | Setup time, read EM1D before EM1OE high | 15 | | ns |
| $t_H$ | EMIF | Data hold time, read EM1D after EM1OE high | 0 | | ns |
| $t_{ELQV}$ | Flash | nCE to Output Delay | | 90 | ns |
| $t_{EHQZ}$ | Flash | nCE High to Output in High Impedance | | 55 | ns |

**Table 25-32. AC Characteristics for a Write Access**

| AC Characteristic | Device | Definition | Minimum | Maximum | Unit |
|---|---|---|---|---|---|
| $t_{AVAV}$ | Flash | Write Cycle Time | 90 | | ns |
| $t_{ELEH}$ | Flash | nCE Pulse Width Low | 50 | | ns |
| $t_{EHEL}$ | Flash | nCE Pulse Width High (not shown in Figure 25-21) | 30 | | ns |

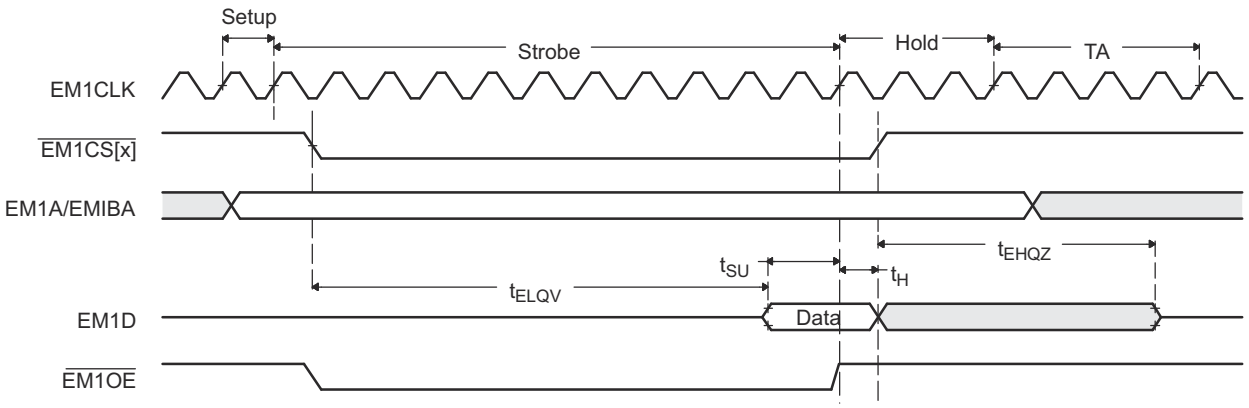



**Figure 25-20. LH28F800BJE-PTTL90 to EMIF Read Timing Waveforms**



Copyright © 2024 Texas Instruments Incorporated

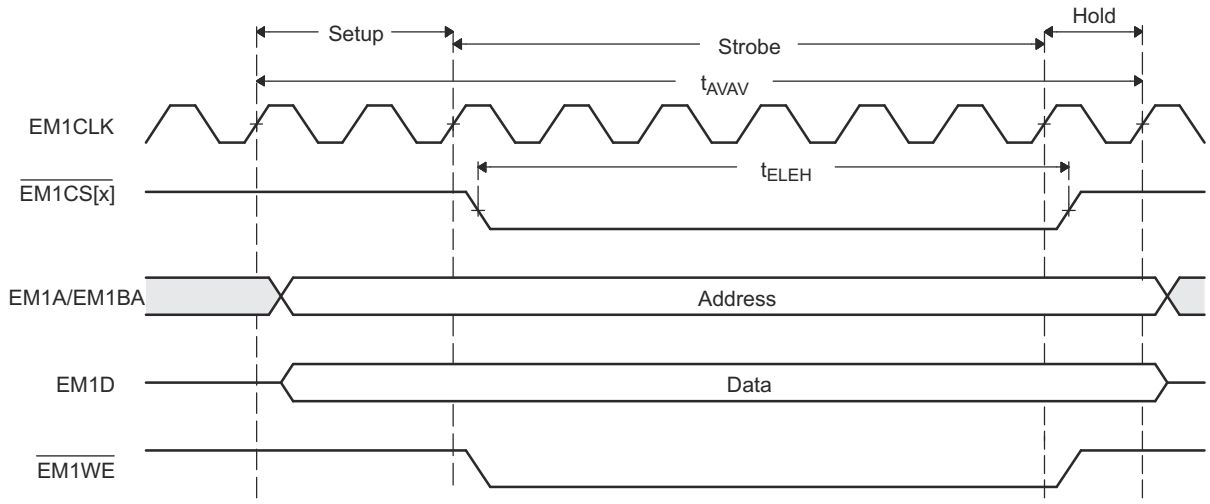



**Figure 25-21. LH28F800BJE-PTTL90 to EMIF Write Timing Waveforms**

The R_STROBE field must be set to meet the following equation:

$R_STROBE >= (t_{ELQV} + t_{SU}) \times f_{EM1CLK} - 1$

R_STROBE >= (90ns + 15ns) × 200MHz - 1

R_STROBE >= 20

R_STROBE = 20

The R_HOLD field must be large enough to satisfy EMIF Data hold time, $t_H$:

$R_HOLD > = t_H \times f_{EM1CLK} - 1$

R_HOLD >= 0ns × 200MHz - 1

R_HOLD >= -1

The R_HOLD field must also combine with the TA field to satisfy the Flash nCE High to Output in High Impedance time, $t_{EHQZ}$:

$R_HOLD + TA >= t_{EHQZ} \times f_{EM1CLK} - 2$

R_HOLD + TA >= 55ns × 200MHz - 2

R_HOLD + TA >= 9

The largest value that can be programmed into the TA field is 3h, therefore the following values must be used:

R_HOLD = 6

TA = 3

For Writes, the W_STROBE field must be set to satisfy the Flash nCE Pulse Width constraint, $t_{ELEH}$:

$W_STROBE >= t_{ELEH} \times f_{EM1CLK} - 1$

W_STROBE >= 50ns × 200MHz - 1

W_STROBE >= 9



Copyright © 2024 Texas Instruments Incorporated

The W_SETUP and W_HOLD fields must combine to satisfy the Flash nCE Pulse Width High constraint, $t_{EHEL}$:

W_SETUP + W_HOLD > = $t_{EHEL}$ × $f_{EM1CLK}$ - 2

W_SETUP + W_HOLD > = 30ns × 200MHz - 2

W_SETUP + W_HOLD > = 4

In addition, the entire Write access length must satisfy the Flash minimum Write Cycle Time, $t_{AVAV}$:

W_SETUP + W_STROBE + W_HOLD >= $t_{AVAV}$ × $f_{EM1CLK}$ - 3

W_SETUP + W_STROBE + W_HOLD >= 90ns × 200MHz - 3

W_SETUP + W_STROBE + W_HOLD >= 15

Solving the above equations for the Write fields results in the following:

W_SETUP = 4

W_STROBE = 10

W_HOLD = 1

Adding a 5ns (1 cycle) margin to each of the periods (excluding TA that is already at the maximum) in this example produces the following recommended values:

W_SETUP = 5h

W_STROBE = Bh

W_HOLD = 2h

R_SETUP = 1h

R_STROBE = 15h

R_HOLD = 7h

TA = 3h



Copyright © 2024 Texas Instruments Incorporated

**Figure 25-22. Asynchronous *m* Configuration Register (*m* = 1, 2) (ASYNC_CS*n*_CR(*n* = 2, 3))**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0101 | | | | 00 | |
| SS | EW | W_SETUP | | | | W_STROBE | |

| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|
| 1011 | | | | 010 | | | 0 |
| W_STROBE | | | | W_HOLD | | | R_SETUP |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 001 | | | 001111 | | | | | | 111 | | | 11 | | 01 | |
| R_SETUP | | | R_STROBE | | | | | | R_HOLD | | | TA | | ASIZE | |

## 25.4 EMIF Registers

This section describes the External Memory Interface Registers.

### *25.4.1 EMIF Base Addresses*

**Table 25-33. EMIF Base Address Table**

| Device Registers | Register Name | Start Address | End Address |
|---|---|---|---|
| Emif1Regs | EMIF_REGS | 0x0004_7000 | 0x0004_77FF |
| Emif2Regs(1) | EMIF_REGS | 0x0004_7800 | 0x0004_7FFF |
| Emif1ConfigRegs | EMIF1_CONFIG_REGS | 0x0005_F480 | 0x0005_F49F |
| Emif2ConfigRegs | EMIF2_CONFIG_REGS | 0x0005_F4A0 | 0x0005_F4BF |

(1) Only available on CPU1.



Copyright © 2024 Texas Instruments Incorporated

### 25.4.2 EMIF_REGS Registers

Table 25-34 lists the memory-mapped registers for the EMIF_REGS registers. All register offset addresses not listed in Table 25-34 should be considered as reserved locations and the register contents should not be modified.

**Table 25-34. EMIF_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | RCSR | Revision Code and Status Register | | Go |
| 2h | ASYNC_WCCR | Async Wait Cycle Config Register | | Go |
| 4h | SDRAM_CR | SDRAM (EMxCS0n) Config Register | | Go |
| 6h | SDRAM_RCR | SDRAM Refresh Control Register | | Go |
| 8h | ASYNC_CS2_CR | Async 1 (EMxCS2n) Config Register | | Go |
| Ah | ASYNC_CS3_CR | Async 2 (EMxCS3n) Config Register | | Go |
| Ch | ASYNC_CS4_CR | Async 3 (EMxCS4n) Config Register | | Go |
| 10h | SDRAM_TR | SDRAM Timing Register | | Go |
| 18h | TOTAL_SDRAM_AR | Total SDRAM Accesses Register | | Go |
| 1Ah | TOTAL_SDRAM_ACTR | Total SDRAM Activate Register | | Go |
| 1Eh | SDR_EXT_TMNG | SDRAM SR/PD Exit Timing Register | | Go |
| 20h | INT_RAW | Interrupt Raw Register | | Go |
| 22h | INT_MSK | Interrupt Masked Register | | Go |
| 24h | INT_MSK_SET | Interrupt Mask Set Register | | Go |
| 26h | INT_MSK_CLR | Interrupt Mask Clear Register | | Go |

Complex bit access types are encoded to fit into small table cells. Table 25-35 shows the codes that are used for access types in this section.

**Table 25-35. EMIF_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| R-0 | R<br>-0 | Read<br>Returns 0s |
| Write Type | | |
| W | W | Write |
| W1S | W<br>1S | Write<br>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. |



Copyright © 2024 Texas Instruments Incorporated

### 25.4.2.1 RCSR Register (Offset = 0h) [Reset = 40000205h]

RCSR is shown in Figure 25-23 and described in Table 25-36.

Return to the Summary Table.

Revision Code and Status Register

**Figure 25-23. RCSR Register**

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

**Table 25-36. RCSR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | BE | R | 0h | EMIF endian mode.<br>0: Little Endian.<br>1: Big Endian.<br>Reset type: SYSRSn |
| 30 | FR | R | 1h | EMIF operating rate.<br>0: Half Rate.<br>1: Full Rate.<br>Reset type: SYSRSn |
| 29-16 | MODULE_ID | R | 0h | EMIF module ID.<br>0x0000: EMIF_24.<br>0x000E: EMIF_24 SDRAM.<br>0x000F: EMIF_24 ASYNC.<br>Reset type: SYSRSn |
| 15-8 | MAJOR_REVISION | R | 2h | Major Revision. EMIF code revisions are indicated by a revision code taking the format major_revision.minor_revision.<br>Reset type: SYSRSn |
| 7-0 | MINOR_REVISION | R | 5h | Minor Revision. See major_revision field description.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 25.4.2.2 ASYNC_WCCR Register (Offset = 2h) [Reset = F0000080h]

ASYNC_WCCR is shown in Figure 25-24 and described in Table 25-37.

Return to the Summary Table.

Async Wait Cycle Config Register

**Figure 25-24. ASYNC_WCCR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | RESERVED | RESERVED | WP0 | RESERVED | | | |
| R/W-1h | R/W-1h | R/W-1h | R/W-1h | R-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 | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MAX_EXT_WAIT | | | | | | | |
| R/W-80h | | | | | | | |

**Table 25-37. ASYNC_WCCR 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 | WP0 | R/W | 1h | Defines the polarity of the EMxWAIT port.:<br>0: Wait if EMxWAIT port is low.<br>1: Wait if EMxWAIT port is high.<br>Reset type: SYSRSn |
| 27-24 | RESERVED | R | 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 |
| 15-8 | RESERVED | R | 0h | Reserved |
| 7-0 | MAX_EXT_WAIT | R/W | 80h | The EMIF will wait for (max_ext_wait + 1) * 16 clock cycles before an extended asynchronous cycle is terminated.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 25.4.2.3 SDRAM_CR Register (Offset = 4h) [Reset = 00000620h]

SDRAM_CR is shown in Figure 25-25 and described in Table 25-38.

Return to the Summary Table.

SDRAM (EMxCS0n) Config Register

**Figure 25-25. SDRAM_CR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| SR | PD | PDWR | RESERVED | | | RESERVED | |
| R/W-0h | R/W-0h | R/W-0h | R-0h | | | R/W-0h | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | RESERVED | | | RESERVED | RESERVED | | RESERVED |
| R/W-0h | R/W-0h | | | R/W-0h | R/W-0h | | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | NM | RESERVED | RESERVED | CL | | | BIT_11_9_LOCK |
| R-0h | R/W-0h | R/W-0h | R/W-0h | R/W-3h | | | R-0/W1S-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | IBANK | | | RESERVED | PAGESIGE | | |
| R-0h | R/W-2h | | | R/W-0h | R/W-0h | | |

**Table 25-38. SDRAM_CR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | SR | R/W | 0h | Self Refresh. Writing a 1 to this bit will cause connected SDRAM devices to be placed into Self Refresh mode and the EMIF to enter the self refresh state. In this state the EMIF will service all asynchronous memory accesses immediately but any SDRAM access will take at least t_ras + 1 cycles due to the time required for the SDRAM devices to out of Self Refresh mode. If an SDRAM access immediately follows the setting of the sr bit, the access will take t_ras + t_xs + 2 cycles. If both sr and pd bits are set, the EMIF will go into Self Refresh.<br>Reset type: SYSRSn |
| 30 | PD | R/W | 0h | Power Down. Writing a 1 to this bit will cause connected SDRAM devices to be placed into Power Down mode. If both sr and pd bits are set, the EMIF will go into Self Refresh.<br>Reset type: SYSRSn |
| 29 | PDWR | R/W | 0h | Perform refreshes during Power Down. Writing a 1 to this bit will cause the EMIF to exit the power down state and issue an AUTO REFRESH command every time Refresh May level is set.<br>Reset type: SYSRSn |
| 28-26 | RESERVED | R | 0h | Reserved |
| 25-23 | RESERVED | R/W | 0h | Reserved |
| 22-20 | RESERVED | R/W | 0h | Reserved |
| 19 | RESERVED | R/W | 0h | Reserved |
| 18-17 | RESERVED | R/W | 0h | Reserved |
| 16 | RESERVED | R/W | 0h | Reserved |
| 15 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 25-38. SDRAM_CR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 14 | NM | R/W | 0h | Narrow mode. Set to 1 when system bus width to memory bus width is 2:1 for SDR SDRAM. Set to 0 when system bus width to memory bus width is 1:1 for SDR SDRAM.<br>A write to this field will cause the EMIF to start the SDRAM initialization sequence.<br>Reset type: SYSRSn |
| 13 | RESERVED | R/W | 0h | Reserved |
| 12 | RESERVED | R/W | 0h | Reserved |
| 11-9 | CL | R/W | 3h | The value of this field defines the CAS latency to be used when accessing connected SDRAM devices. Only CAS latencies of 2 (cl = 2) and 3 (cl = 3) are supported.<br>A write to this field will cause the EMIF to start the SDRAM initialization sequence.<br>Reset type: SYSRSn |
| 8 | BIT_11_9_LOCK | R-0/W1S | 0h | Bits 11 to 9 can only be written if this bit is set to 1.<br>Reset type: SYSRSn |
| 7 | RESERVED | R | 0h | Reserved |
| 6-4 | IBANK | R/W | 2h | Defines number of banks inside connected SDRAM devices:<br>000: 1 bank SDRAM devices.<br>001: 2 bank SDRAM devices.<br>010: 4 bank SDRAM devices.<br>011: Reserved.<br>1xx: Reserved.<br>A write to this field will cause the EMIF to start the SDRAM initialization sequence.<br>Reset type: SYSRSn |
| 3 | RESERVED | R/W | 0h | Reserved |
| 2-0 | PAGESIGE | R/W | 0h | Defines the internal page size of connected SDRAM devices:<br>000: 256-word pages requiring 8 column address bits.<br>001: 512-word pages requiring 9 column address bits.<br>010: 1024-word pages requiring 10 column address bits.<br>011: 2048-word pages requiring 11 column address bits.<br>1xx: Reserved.<br>A write to this field will cause the EMIF to start the SDRAM initialization sequence.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 25.4.2.4 SDRAM_RCR Register (Offset = 6h) [Reset = 00000080h]

SDRAM_RCR is shown in Figure 25-26 and described in Table 25-39.

Return to the Summary Table.

SDRAM Refresh Control Register

**Figure 25-26. SDRAM_RCR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | RESERVED | | |
| R-0h | | | | | R/W-0h | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | REFRESH_RATE | | | | |
| R-0h | | | R/W-80h | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| REFRESH_RATE | | | | | | | |
| R/W-80h | | | | | | | |

**Table 25-39. SDRAM_RCR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-19 | RESERVED | R | 0h | Reserved |
| 18-16 | RESERVED | R/W | 0h | Reserved |
| 15-13 | RESERVED | R | 0h | Reserved |
| 12-0 | REFRESH_RATE | R/W | 80h | Value in this field is used to define the rate at which connected SDRAM devices will be refreshed, as follows:<br>SDRAM refresh rate = EMIF rate/refresh_rate<br>where EMIF rate=clk rate when full_rate=1, or EMIF rate=1/2 clk rate when full_rate=0.<br>Writing a value < 0x0020 to this field will cause it to be loaded with (2 * t_rfc) + 1 value from SDRAM Timing register.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 25.4.2.5 ASYNC_CS2_CR Register (Offset = 8h) [Reset = 3FFFFFFDh]

ASYNC_CS2_CR is shown in Figure 25-27 and described in Table 25-40.

Return to the Summary Table.

Async 1 (EMxCS2n) Config Register

**Figure 25-27. ASYNC_CS2_CR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SS | EW | W_SETUP | | | | W_STROBE | | | | | | W_HOLD | | | R_SETUP |
| R/W-0h | R/W-0h | R/W-Fh | | | | R/W-3Fh | | | | | | R/W-7h | | | R/W-Fh |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R_SETUP | | | R_STROBE | | | | | | R_HOLD | | | TA | | ASIZE | |
| R/W-Fh | | | R/W-3Fh | | | | | | R/W-7h | | | R/W-3h | | R/W-1h | |

**Table 25-40. ASYNC_CS2_CR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | SS | R/W | 0h | Select Strobe mode. Set to 1 if chip selects need to have write or read strobe timing.<br>Reset type: SYSRSn |
| 30 | EW | R/W | 0h | Extend Wait mode. Set to 1 if extended asynchronous cycles are required based on EMxWAIT.<br>Reset type: SYSRSn |
| 29-26 | W_SETUP | R/W | Fh | Write Strobe Setup cycles. Number of EMxCLK cycles from EMxAy, EMxBAy, EMxDQMy, and EMxCS2n being set to EMxWEn asserted, minus one cycle. The reset value is 16 cycles.<br>Reset type: SYSRSn |
| 25-20 | W_STROBE | R/W | 3Fh | Write Strobe Duration cycles. Number of EMxCLK cycles for which EMxWEn is held active, minus one cycle. The reset value is 64 cycles. This field cannot be zero when ew = 1.<br>Reset type: SYSRSn |
| 19-17 | W_HOLD | R/W | 7h | Write Strobe Hold cycles. Number of EMxCLK cycles for which EMxAy, EMxBAy, EMxDQMy, and EMxCS2n are held after EMxWEn has been deasserted, minus one cycle. The reset value is 8 cycles.<br>Reset type: SYSRSn |
| 16-13 | R_SETUP | R/W | Fh | Read Strobe Setup cycles. Number of EMxCLK cycles from EMxAy, EMxBAy, EMxDQMy, and EMxCS2n being set to EMxOEn asserted, minus one cycle. The reset value is 16 cycles.<br>Reset type: SYSRSn |
| 12-7 | R_STROBE | R/W | 3Fh | Read Strobe Duration cycles. Number of EMxCLK cycles for which EMxOEn is held active, minus one cycle. The reset value is 64 cycles. This field cannot be zero when ew = 1.<br>Reset type: SYSRSn |
| 6-4 | R_HOLD | R/W | 7h | Read Strobe Hold cycles. Number of EMxCLK cycles for which EMxAy, EMxBAy, EMxDQMy, and EMxCS2n are held after EMxOEn has been deasserted, minus one cycle. The reset value is 8 cycles.<br>Reset type: SYSRSn |
| 3-2 | TA | R/W | 3h | Turn Around cycles. Number of EMxCLK cycles between the end of one asynchronous memory access and the start of another asynchronous memory access, minus one cycle. This delay is not incurred between a read followed by a read, or a write followed by a write to the same chip select. The reset value is 4 cycles.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 25-40. ASYNC_CS2_CR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1-0 | ASIZE | R/W | 1h | Asynchronous Memory Size. Defines the width of the asynchronous device's data bus :<br>00: 8 Bit data bus.<br>01: 16 Bit data bus.<br>10: 32 Bit data bus.<br>11: Reserved.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 25.4.2.6 ASYNC_CS3_CR Register (Offset = Ah) [Reset = 3FFFFFFDh]

ASYNC_CS3_CR is shown in Figure 25-28 and described in Table 25-41.

Return to the Summary Table.

Async 2 (EMxCS3n) Config Register

**Figure 25-28. ASYNC_CS3_CR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SS | EW | W_SETUP | | | | W_STROBE | | | | | | W_HOLD | | | R_SETUP |
| R/W-0h | R/W-0h | R/W-Fh | | | | R/W-3Fh | | | | | | R/W-7h | | | R/W-Fh |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R_SETUP | | | R_STROBE | | | | | | R_HOLD | | | TA | | ASIZE | |
| R/W-Fh | | | R/W-3Fh | | | | | | R/W-7h | | | R/W-3h | | R/W-1h | |

**Table 25-41. ASYNC_CS3_CR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | SS | R/W | 0h | Select Strobe mode. Set to 1 if chip selects need to have write or read strobe timing.<br>Reset type: SYSRSn |
| 30 | EW | R/W | 0h | Extend Wait mode. Set to 1 if extended asynchronous cycles are required based on EMxWAIT.<br>Reset type: SYSRSn |
| 29-26 | W_SETUP | R/W | Fh | Write Strobe Setup cycles. Number of EMxCLK cycles from EMxAy, EMxBAy, EMxDQMy, and EMxCS3n being set to EMxWEn asserted, minus one cycle. The reset value is 16 cycles.<br>Reset type: SYSRSn |
| 25-20 | W_STROBE | R/W | 3Fh | Write Strobe Duration cycles. Number of EMxCLK cycles for which EMxWEn is held active, minus one cycle. The reset value is 64 cycles. This field cannot be zero when ew = 1.<br>Reset type: SYSRSn |
| 19-17 | W_HOLD | R/W | 7h | Write Strobe Hold cycles. Number of EMxCLK cycles for which EMxAy, EMxBAy, EMxDQMy, and EMxCS3n are held after EMxWEn has been deasserted, minus one cycle. The reset value is 8 cycles.<br>Reset type: SYSRSn |
| 16-13 | R_SETUP | R/W | Fh | Read Strobe Setup cycles. Number of EMxCLK cycles from EMxAy, EMxBAy, EMxDQMy, and EMxCS3n being set to EMxOEn asserted, minus one cycle. The reset value is 16 cycles.<br>Reset type: SYSRSn |
| 12-7 | R_STROBE | R/W | 3Fh | Read Strobe Duration cycles. Number of EMxCLK cycles for which EMxOEn is held active, minus one cycle. The reset value is 64 cycles. This field cannot be zero when ew = 1.<br>Reset type: SYSRSn |
| 6-4 | R_HOLD | R/W | 7h | Read Strobe Hold cycles. Number of EMxCLK cycles for which EMxAy, EMxBAy, EMxDQMy, and EMxCS3n are held after EMxOEn has been deasserted, minus one cycle. The reset value is 8 cycles.<br>Reset type: SYSRSn |
| 3-2 | TA | R/W | 3h | Turn Around cycles. Number of EMxCLK cycles between the end of one asynchronous memory access and the start of another asynchronous memory access, minus one cycle. This delay is not incurred between a read followed by a read, or a write followed by a write to the same chip select. The reset value is 4 cycles.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 25-41. ASYNC_CS3_CR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1-0 | ASIZE | R/W | 1h | Asynchronous Memory Size. Defines the width of the asynchronous device's data bus :<br>00: 8 Bit data bus.<br>01: 16 Bit data bus.<br>10: 32 Bit data bus.<br>11: Reserved.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 25.4.2.7 ASYNC_CS4_CR Register (Offset = Ch) [Reset = 3FFFFFFDh]

ASYNC_CS4_CR is shown in Figure 25-29 and described in Table 25-42.

Return to the Summary Table.

Async 3 (EMxCS4n) Config Register

**Figure 25-29. ASYNC_CS4_CR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SS | EW | W_SETUP | | | | W_STROBE | | | | | | W_HOLD | | | R_SETUP |
| R/W-0h | R/W-0h | R/W-Fh | | | | R/W-3Fh | | | | | | R/W-7h | | | R/W-Fh |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R_SETUP | | | R_STROBE | | | | | | R_HOLD | | | TA | | ASIZE | |
| R/W-Fh | | | R/W-3Fh | | | | | | R/W-7h | | | R/W-3h | | R/W-1h | |

**Table 25-42. ASYNC_CS4_CR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | SS | R/W | 0h | Select Strobe mode. Set to 1 if chip selects need to have write or read strobe timing.<br>Reset type: SYSRSn |
| 30 | EW | R/W | 0h | Extend Wait mode. Set to 1 if extended asynchronous cycles are required based on EMxWAIT.<br>Reset type: SYSRSn |
| 29-26 | W_SETUP | R/W | Fh | Write Strobe Setup cycles. Number of EMxCLK cycles from EMxAy, EMxBAy, EMxDQMy, and EMxCS4n being set to EMxWEn asserted, minus one cycle. The reset value is 16 cycles.<br>Reset type: SYSRSn |
| 25-20 | W_STROBE | R/W | 3Fh | Write Strobe Duration cycles. Number of EMxCLK cycles for which EMxWEn is held active, minus one cycle. The reset value is 64 cycles. This field cannot be zero when ew = 1.<br>Reset type: SYSRSn |
| 19-17 | W_HOLD | R/W | 7h | Write Strobe Hold cycles. Number of EMxCLK cycles for which EMxAy, EMxBAy, EMxDQMy, and EMxCS4n are held after EMxWEn has been deasserted, minus one cycle. The reset value is 8 cycles.<br>Reset type: SYSRSn |
| 16-13 | R_SETUP | R/W | Fh | Read Strobe Setup cycles. Number of EMxCLK cycles from EMxAy, EMxBAy, EMxDQMy, and EMxCS4n being set to EMxOEn asserted, minus one cycle. The reset value is 16 cycles.<br>Reset type: SYSRSn |
| 12-7 | R_STROBE | R/W | 3Fh | Read Strobe Duration cycles. Number of EMxCLK cycles for which EMxOEn is held active, minus one cycle. The reset value is 64 cycles. This field cannot be zero when ew = 1.<br>Reset type: SYSRSn |
| 6-4 | R_HOLD | R/W | 7h | Read Strobe Hold cycles. Number of EMxCLK cycles for which EMxAy, EMxBAy, EMxDQMy, and EMxCS4n are held after EMxOEn has been deasserted, minus one cycle. The reset value is 8 cycles.<br>Reset type: SYSRSn |
| 3-2 | TA | R/W | 3h | Turn Around cycles. Number of EMxCLK cycles between the end of one asynchronous memory access and the start of another asynchronous memory access, minus one cycle. This delay is not incurred between a read followed by a read, or a write followed by a write to the same chip select. The reset value is 4 cycles.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 25-42. ASYNC_CS4_CR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1-0 | ASIZE | R/W | 1h | Asynchronous Memory Size. Defines the width of the asynchronous device's data bus :<br>00: 8 Bit data bus.<br>01: 16 Bit data bus.<br>10: 32 Bit data bus.<br>11: Reserved.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 25.4.2.8 SDRAM_TR Register (Offset = 10h) [Reset = 19214610h]

SDRAM_TR is shown in Figure 25-30 and described in Table 25-43.

Return to the Summary Table.

SDRAM Timing Register

**Figure 25-30. SDRAM_TR Register**

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

**Table 25-43. SDRAM_TR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-27 | T_RFC | R/W | 3h | Minimum number of EMxCLK cycles from Refresh or Load Mode to Refresh or Activate, minus one.<br>Reset type: SYSRSn |
| 26-24 | T_RP | R/W | 1h | Minimum number of EMxCLK cycles from Precharge to Activate or Refresh, minus one.<br>Reset type: SYSRSn |
| 23 | RESERVED | R | 0h | Reserved |
| 22-20 | T_RCD | R/W | 2h | Minimum number of EMxCLK cycles from Activate to Read or Write, minus one.<br>Reset type: SYSRSn |
| 19 | RESERVED | R | 0h | Reserved |
| 18-16 | T_WR | R/W | 1h | For SDR, this is equal to minimum number of EMxCLK cycles from last Write transfer to Precharge, minus one.<br>Reset type: SYSRSn |
| 15-12 | T_RAS | R/W | 4h | Minimum number of EMxCLK cycles from Activate to Precharge, minus one. t_ras >= t_rcd.<br>Reset type: SYSRSn |
| 11-8 | T_RC | R/W | 6h | Minimum number of EMxCLK cycles from Activate to Activate minus one.<br>Reset type: SYSRSn |
| 7 | RESERVED | R | 0h | Reserved |
| 6-4 | T_RRD | R/W | 1h | Minimum number of EMxCLK cycles from Activate to Activate for a different bank, minus one.<br>Reset type: SYSRSn |
| 3-0 | RESERVED | R | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 25.4.2.9 TOTAL_SDRAM_AR Register (Offset = 18h) [Reset = 00000000h]

TOTAL_SDRAM_AR is shown in Figure 25-31 and described in Table 25-44.

Return to the Summary Table.

Total SDRAM Accesses Register

**Figure 25-31. TOTAL_SDRAM_AR Register**

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

**Table 25-44. TOTAL_SDRAM_AR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | TOTAL_SDRAM_AR | R | 0h | Indicates the total number of accesses to SDRAM from a master (CPUx/CPUX.DMA). This counter is incremented by two for a single access crossing page boundaries.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 25.4.2.10 TOTAL_SDRAM_ACTR Register (Offset = 1Ah) [Reset = 00000000h]

TOTAL_SDRAM_ACTR is shown in Figure 25-32 and described in Table 25-45.

Return to the Summary Table.

Total SDRAM Activate Register

**Figure 25-32. TOTAL_SDRAM_ACTR Register**

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

**Table 25-45. TOTAL_SDRAM_ACTR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | TOTAL_SDRAM_ACTR | R | 0h | Indicates the total number of SDRAM accesses which require an activate command.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 25.4.2.11 SDR_EXT_TMNG Register (Offset = 1Eh) [Reset = 00000007h]

SDR_EXT_TMNG is shown in Figure 25-33 and described in Table 25-46.

Return to the Summary Table.

SDRAM SR/PD Exit Timing Register

**Figure 25-33. SDR_EXT_TMNG Register**

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

**Table 25-46. SDR_EXT_TMNG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-5 | RESERVED | R | 0h | Reserved |
| 4-0 | T_XS | R/W | 7h | This is equal to minimum number of EMxCLK cycles from Self Refresh exit to any command, minus one. For SDR SDRAM, this count should satisfy tXSR.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 25.4.2.12 INT_RAW Register (Offset = 20h) [Reset = 00000000h]

INT_RAW is shown in Figure 25-34 and described in Table 25-47.

Return to the Summary Table.

Interrupt Raw Register

**Figure 25-34. INT_RAW Register**

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

**Table 25-47. INT_RAW Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-6 | RESERVED | R | 0h | Reserved |
| 5-2 | WR | R/W1S | 0h | Wait Rise. Set to 1 by hardware to indicate rising edge on the corresponding EMxWAIT has been detected. The WPx bits in the Async Wait Cycle Config register has no effect on these bits. Writing a 1 will clear these bits as well as the wr_masked bits in the Interrupt Masked register. Writing a 0 has no effect.<br>Reset type: SYSRSn |
| 1 | LT | R/W1S | 0h | Line Trap. Set to 1 by hardware to indicate illegal memory access type or invalid cache line size. Writing a 1 will clear this bit as well as the lt_masked bit in the Interrupt Masked register. Writing a 0 has no effect.<br>Reset type: SYSRSn |
| 0 | AT | R/W1S | 0h | Asynchronous Timeout. Set to 1 by hardware to indicate that during an extended asynchronous memory access cycle, the EMxWAIT signal did not go inactive within the number of cycles defined by the max_ext_wait field in Async Wait Cycle Config register. Writing a 1 will clear this bit as well as the at_masked bit in the Interrupt Masked register. Writing a 0 has no effect.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 25.4.2.13 INT_MSK Register (Offset = 22h) [Reset = 00000000h]

INT_MSK is shown in Figure 25-35 and described in Table 25-48.

Return to the Summary Table.

Interrupt Masked Register

**Figure 25-35. INT_MSK Register**

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

**Table 25-48. INT_MSK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-6 | RESERVED | R | 0h | Reserved |
| 5-2 | WR_MASKED | R/W1S | 0h | Masked Wait Rise. Set to 1 by hardware to indicate rising edge on the corresponding EMxWAIT has been detected, only if the wr_mask_set bit in the Interrupt Mask Set register is set to 1. The WPx bits in the Async Wait Cycle Config register has no effect on these bits. Writing a 1 will clear these bits as well as the wr bits in the Interrupt Raw register. Writing a 0 has no effect.<br>Reset type: SYSRSn |
| 1 | LT_MASKED | R/W1S | 0h | Masked Line Trap. Set to 1 by hardware to indicate illegal memory access type or invalid cache line size, only if the lt_mask_set bit in the Interrupt Mask Set register is set to 1. Writing a 1 will clear this bit as well as the lt bit in the Interrupt Raw register. Writing a 0 has no effect.<br>Reset type: SYSRSn |
| 0 | AT_MASKED | R/W1S | 0h | Masked Asynchronous Timeout. Set to 1 by hardware to indicate that during an extended asynchronous memory access cycle, the EMxWAIT signal did not go inactive within the number of cycles defined by the max_ext_wait field in Async Wait Cycle Config register, only if the at_mask_set bit in the Interrupt Mask Set register is set to 1. Writing a 1 will clear this bit as well as the at bit in the Interrupt Raw register. Writing a 0 has no effect.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 25.4.2.14 INT_MSK_SET Register (Offset = 24h) [Reset = 00000000h]

INT_MSK_SET is shown in Figure 25-36 and described in Table 25-49.

Return to the Summary Table.

Interrupt Mask Set Register

**Figure 25-36. INT_MSK_SET Register**

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

**Table 25-49. INT_MSK_SET Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-6 | RESERVED | R | 0h | Reserved |
| 5-2 | WR_MASK_SET | R/W1S | 0h | Mask set for wr_masked bits in the Interrupt Masked Register. Writing a 1 will enable the interrupts, and set these bits as well as the wr_mask_clr bits in the Interrupt Mask Clear register. Writing a 0 has no effect.<br>Reset type: SYSRSn |
| 1 | LT_MASK_SET | R/W1S | 0h | Mask set for lt_masked bit in the Interrupt Masked Register. Writing a 1 will enable the interrupt, and set this bit as well as the lt_mask_clr bit in the Interrupt Mask Clear register. Writing a 0 has no effect.<br>Reset type: SYSRSn |
| 0 | AT_MASK_SET | R/W1S | 0h | Mask set for at_masked bit in the Interrupt Masked Register. Writing a 1 will enable the interrupt, and set this bit as well as the at_mask_clr bit in the Interrupt Mask Clear register. Writing a 0 has no effect.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 25.4.2.15 INT_MSK_CLR Register (Offset = 26h) [Reset = 00000000h]

INT_MSK_CLR is shown in Figure 25-37 and described in Table 25-50.

Return to the Summary Table.

Interrupt Mask Clear Register

**Figure 25-37. INT_MSK_CLR Register**

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

**Table 25-50. INT_MSK_CLR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-6 | RESERVED | R | 0h | Reserved |
| 5-2 | WR_MASK_CLR | R/W1S | 0h | Mask clear for wr_masked bits in the Interrupt Masked Register. Writing a 1 will disable the interrupts, and clear these bits as well as the wr_mask_set bits in the Interrupt Mask Set register. Writing a 0 has no effect.<br>Reset type: SYSRSn |
| 1 | LT_MASK_CLR | R/W1S | 0h | Mask clear for lt_masked bit in the Interrupt Masked Register. Writing a 1 will disable the interrupt, and clear this bit as well as the lt_mask_set bit in the Interrupt Mask Set register. Writing a 0 has no effect.<br>Reset type: SYSRSn |
| 0 | AT_MASK_CLR | R/W1S | 0h | Mask clear for at_masked bit in the Interrupt Masked Register. Writing a 1 will disable the interrupt, and clear this bit as well as the at_mask_set bit in the Interrupt Mask Set register. Writing a 0 has no effect.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 25.4.3 EMIF1_CONFIG_REGS Registers

Table 25-51 lists the memory-mapped registers for the EMIF1_CONFIG_REGS registers. All register offset addresses not listed in Table 25-51 should be considered as reserved locations and the register contents should not be modified.

**Table 25-51. EMIF1_CONFIG_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | EMIF1LOCK | EMIF1 Config Lock Register | EALLOW | Go |
| 2h | EMIF1COMMIT | EMIF1 Config Lock Commit Register | EALLOW | Go |
| 4h | EMIF1MSEL | EMIF1 Master Sel Register | EALLOW | Go |
| 8h | EMIF1ACCPROT0 | EMIF1 Config Register 0 | EALLOW | Go |

Complex bit access types are encoded to fit into small table cells. Table 25-52 shows the codes that are used for access types in this section.

**Table 25-52. EMIF1_CONFIG_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| R-0 | R<br>-0 | Read<br>Returns 0s |
| Write Type | | |
| W | W | Write |
| WSonce | W<br>Sonce | Write<br>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. |



Copyright © 2024 Texas Instruments Incorporated

### 25.4.3.1 EMIF1LOCK Register (Offset = 0h) [Reset = 00000000h]

EMIF1LOCK is shown in Figure 25-38 and described in Table 25-53.

Return to the Summary Table.

EMIF1 Config Lock Register

**Figure 25-38. EMIF1LOCK Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | | LOCK_EMIF1 |
| R-0h | | | | | | | R/W-0h |

**Table 25-53. EMIF1LOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-1 | RESERVED | R | 0h | Reserved |
| 0 | LOCK_EMIF1 | R/W | 0h | Locks the write to access protection and master select fields for EMIF1:<br>0: Write to ACCPROT and Mselect fields are allowed.<br>1: Write to ACCPROT and Mselect fields are blocked.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 25.4.3.2 EMIF1COMMIT Register (Offset = 2h) [Reset = 00000000h]

EMIF1COMMIT is shown in Figure 25-39 and described in Table 25-54.

Return to the Summary Table.

EMIF1 Config Lock Commit Register

**Figure 25-39. EMIF1COMMIT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | | COMMIT_EMIF1 |
| R-0h | | | | | | | R/WSonce-0h |

**Table 25-54. EMIF1COMMIT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-1 | RESERVED | R | 0h | Reserved |
| 0 | COMMIT_EMIF1 | R/WSonce | 0h | Permanently Locks the write to access protection and master select fields for EMIF1:<br>0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in EMIF1LOCK register.<br>1: Write to ACCPROT and Mselect fields are permanently blocked.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 25.4.3.3 EMIF1MSEL Register (Offset = 4h) [Reset = 00000000h]

EMIF1MSEL is shown in Figure 25-40 and described in Table 25-55.

Return to the Summary Table.

EMIF1 Master Sel Register

**Figure 25-40. EMIF1MSEL Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| KEY | | | | | | | |
| R-0/W-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| KEY | | | | | | | |
| R-0/W-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| KEY | | | | | | | |
| R-0/W-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| KEY | | | | RESERVED | | MSEL_EMIF1 | |
| R-0/W-0h | | | | R-0h | | R/W-0h | |

**Table 25-55. EMIF1MSEL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-4 | KEY | R-0/W | 0h | Writing the value 0x93A5CE7 will allow the writing of the EMIF1Mselect bits, else writes are ignored. Reads will return 0.<br>Reset type: CPU1.SYSRSn |
| 3-2 | RESERVED | R | 0h | Reserved |
| 1-0 | MSEL_EMIF1 | R/W | 0h | Master Select for EMIF1:<br>00: CPU1 is master but not grabbed. CPU2 can grab the master ownership by changing this value to '10'.<br>01: CPU1 is master.<br>10: CPU2 is master.<br>11: CPU1 is master but not grabbed. CPU2 can grab the master ownership by changing this value to '10'.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 25.4.3.4 EMIF1ACCPROT0 Register (Offset = 8h) [Reset = 00000000h]

EMIF1ACCPROT0 is shown in Figure 25-41 and described in Table 25-56.

Return to the Summary Table.

EMIF1 Config Register 0

**Figure 25-41. EMIF1ACCPROT0 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | DMAWRPROT_EMIF1 | CPUWRPROT_EMIF1 | FETCHPROT_EMIF1 |
| R-0h | | | | | R/W-0h | R/W-0h | R/W-0h |

**Table 25-56. EMIF1ACCPROT0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-3 | RESERVED | R | 0h | Reserved |
| 2 | DMAWRPROT_EMIF1 | R/W | 0h | DMA WR Protection For EMIF1:<br>0: DMA Writes are allowed.<br>1: DMA Writes are blocked.<br>Reset type: SYSRSn |
| 1 | CPUWRPROT_EMIF1 | R/W | 0h | CPU WR Protection For EMIF1:<br>0: CPU Writes are allowed.<br>1: CPU Writes are blocked.<br>Reset type: SYSRSn |
| 0 | FETCHPROT_EMIF1 | R/W | 0h | Fetch Protection For EMIF1:<br>0: CPU Fetches are allowed.<br>1: CPU Fetches are blocked.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 25.4.4 EMIF2_CONFIG_REGS Registers

Table 25-57 lists the memory-mapped registers for the EMIF2_CONFIG_REGS registers. All register offset addresses not listed in Table 25-57 should be considered as reserved locations and the register contents should not be modified.

**Table 25-57. EMIF2_CONFIG_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | EMIF2LOCK | EMIF2 Config Lock Register | | Go |
| 2h | EMIF2COMMIT | EMIF2 Config Lock Commit Register | EALLOW | Go |
| 8h | EMIF2ACCPROT0 | EMIF2 Config Register 0 | EALLOW | Go |

Complex bit access types are encoded to fit into small table cells. Table 25-58 shows the codes that are used for access types in this section.

**Table 25-58. EMIF2_CONFIG_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| Write Type | | |
| W | W | Write |
| WSonce | W<br>Sonce | Write<br>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. |



Copyright © 2024 Texas Instruments Incorporated

### 25.4.4.1 EMIF2LOCK Register (Offset = 0h) [Reset = 00000000h]

EMIF2LOCK is shown in Figure 25-42 and described in Table 25-59.

Return to the Summary Table.

EMIF2 Config Lock Register

**Figure 25-42. EMIF2LOCK Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | | LOCK_EMIF2 |
| R-0h | | | | | | | R/W-0h |

**Table 25-59. EMIF2LOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-1 | RESERVED | R | 0h | Reserved |
| 0 | LOCK_EMIF2 | R/W | 0h | Locks the write to access protection fields for EMIF2:<br>0: Write to ACCPROT and Mselect fields are allowed.<br>1: Write to ACCPROT and Mselect fields are blocked.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 25.4.4.2 EMIF2COMMIT Register (Offset = 2h) [Reset = 00000000h]

EMIF2COMMIT is shown in Figure 25-43 and described in Table 25-60.

Return to the Summary Table.

EMIF2 Config Lock Commit Register

**Figure 25-43. EMIF2COMMIT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | | COMMIT_EMIF2 |
| R-0h | | | | | | | R/WSonce-0h |

**Table 25-60. EMIF2COMMIT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-1 | RESERVED | R | 0h | Reserved |
| 0 | COMMIT_EMIF2 | R/WSonce | 0h | Permanently Locks the write to access protection fields for EMIF2:<br>0: Write to ACCPROT fields are allowed based on value of lock field in EMIF2LOCK register.<br>1: Write to ACCPROT fields are permanently blocked.<br>Reset type: CPU1.SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 25.4.4.3 EMIF2ACCPROT0 Register (Offset = 8h) [Reset = 00000000h]

EMIF2ACCPROT0 is shown in Figure 25-44 and described in Table 25-61.

Return to the Summary Table.

EMIF2 Config Register 0

**Figure 25-44. EMIF2ACCPROT0 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | CPUWRPROT_EMIF2 | FETCHPROT_EMIF2 |
| R-0h | | | | | | R/W-0h | R/W-0h |

**Table 25-61. EMIF2ACCPROT0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-2 | RESERVED | R | 0h | Reserved |
| 1 | CPUWRPROT_EMIF2 | R/W | 0h | CPU WR Protection For EMIF2:<br>0: CPU Writes are allowed.<br>1: CPU Writes are blocked.<br>Reset type: SYSRSn |
| 0 | FETCHPROT_EMIF2 | R/W | 0h | Fetch Protection For EMIF2<br>0: CPU Fetches are allowed.<br>1: CPU Fetches are blocked.<br>Reset type: SYSRSn |

### *25.4.5 EMIF Registers to Driverlib Functions*

**Table 25-62. EMIF Registers to Driverlib Functions**

| File | Driverlib Function |
|---|---|
| **RCSR** | |
| - | |
| **ASYNC_WCCR** | |
| emif.h | EMIF_setAsyncWaitPolarity |
| emif.h | EMIF_setAsyncMaximumWaitCycles |
| **SDRAM_CR** | |
| emif.h | EMIF_setSyncMemoryConfig |
| emif.h | EMIF_enableSyncSelfRefresh |
| emif.h | EMIF_disableSyncSelfRefresh |
| emif.h | EMIF_enableSyncPowerDown |
| emif.h | EMIF_disableSyncPowerDown |
| emif.h | EMIF_enableSyncRefreshInPowerDown |



Copyright © 2024 Texas Instruments Incorporated

**Table 25-62. EMIF Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| emif.h | EMIF_disableSyncRefreshInPowerDown |
| **SDRAM_RCR** | |
| emif.h | EMIF_setSyncRefreshRate |
| **ASYNC_CS2_CR** | |
| emif.h | EMIF_setAsyncMode |
| emif.h | EMIF_enableAsyncExtendedWait |
| emif.h | EMIF_setAsyncTimingParams |
| emif.h | EMIF_setAsyncDataBusWidth |
| **ASYNC_CS3_CR** | |
| - | See ASYNC_CS2_CR |
| **ASYNC_CS4_CR** | |
| - | See ASYNC_CS2_CR |
| **SDRAM_TR** | |
| emif.h | EMIF_setSyncTimingParams |
| **TOTAL_SDRAM_AR** | |
| emif.h | EMIF_getSyncTotalAccesses |
| **TOTAL_SDRAM_ACTR** | |
| emif.h | EMIF_getSyncTotalActivateAccesses |
| **SDR_EXT_TMNG** | |
| emif.h | EMIF_setSyncSelfRefreshExitTmng |
| **INT_RAW** | |
| - | |
| **INT_MSK** | |
| emif.h | EMIF_enableAsyncInterrupt |
| emif.h | EMIF_disableAsyncInterrupt |
| emif.h | EMIF_getAsyncInterruptStatus |
| emif.h | EMIF_clearAsyncInterruptStatus |
| **INT_MSK_SET** | |
| emif.h | EMIF_enableAsyncInterrupt |
| **INT_MSK_CLR** | |
| emif.h | EMIF_disableAsyncInterrupt |



Copyright © 2024 Texas Instruments Incorporated

Chapter 26

# Configurable Logic Block (CLB)

This chapter describes the features and operation of the configurable logic block (CLB) that is a collection of configurable blocks that can be inter-connected using software to implement custom digital logic functions.


**26.1 Introduction** ..... 2876
**26.2 Description** ..... 2876
**26.3 CLB Input/Output Connection** ..... 2879
**26.4 CLB Tile** ..... 2887
**26.5 CPU Interface** ..... 2900
**26.6 DMA Access** ..... 2901
**26.7 Software** ..... 2902
**26.8 CLB Registers** ..... 2905





Copyright © 2024 Texas Instruments Incorporated

## 26.1 Introduction

The configurable logic block (CLB) is a collection of configurable blocks that can be inter-connected using software to implement custom digital logic functions. The CLB is able to enhance existing peripherals through a set of crossbar interconnections, which provide a high level of connectivity to existing control peripherals such as enhanced pulse width modulators (ePWM), enhanced capture modules (eCAP), and enhanced quadrature encoder pulse modules (eQEP). The crossbars also allow the CLB to be connected to external GPIO pins. In this way, the CLB can be configured to interact with device peripherals to perform small logical functions such as simple PWM generators, or to implement custom serial data exchange protocols.

The CLB peripheral is configured through the CLB tool. For more information on the CLB tool, available examples, application reports, and user's guide, refer to the following location in your C2000WARE package (C2000Ware_2_00_00_03 and higher): C2000WARE_INSTALL_LOCATION\utilities\clb_tool\clb_syscfg\doc

### *26.1.1 CLB Related Collateral*

**Foundational Materials**

- C2000 Academy - CLB
- C2000™ Configurable Logic Block (CLB) Series (Video)
- Customizing on-chip peripherals defies conventional logic
- *Enable Differentiation and win with CLB in various applications Application Report*
- Enable Differentiation with Configurable Logic in Various Automotive Applications (Video)

**Getting Started Materials**

- *C2000™ Position Manager PTO API Reference Guide Application Report*
- *CLB Tool User Guide*
  - Basic examples are 7 - 15 (start with these). More involved examples are 1-6.
- *Designing With The C2000 Configurable Logic Block Application Report*
- How do I add SYSCONFIG support (Pinmux and Peripheral Initialization) to an existing driverlib project?
- *How to Migrate Custom Logic From an FPGA/CPLD to C2000 Microcontrollers Application Report*
  - Chpaters 1-3 are very useful for getting started and learning the CLB. Later chapters are very useful Expert materials for migrating from FPGA/CPLD to C2000's CLB.

**Expert Materials**

- *Achieve Delayed Protection for Three-Level Inverter With CLB Application Report*
- *Diagnosing Delta-Sigma Modulator Bitstream Using C2000™ Configurable Logic Block Application Report*
- *How to Implement Custom Serial Interfaces Using Configurable Logic Block (CLB) Application Report*
- *Tamagawa T-Format Absolute-Encoder Master Interface Reference Design for C2000™ MCUs*

## 26.2 Description

The CLB subsystem contains a number of identical tiles. There are four such tiles in the CLB subsystem; other devices can contain more or fewer tiles. Tiles are numbered 1 to N, where N is the total tile count on the device. Each tile contains combinational and sequential logic blocks, as well as other dedicated hardware to be described later in this document. Figure 26-1 shows the structure of the CLB subsystem in the device.

The tile contains the core logic, providing the logic reconfiguration capability. Each CLB tile is associated with a separate CPU interface, which contains the registers needed to control and configure the logic in the tile. The CPU interface also contains data transfer buffers that can be used as part of the configurable logic to exchange data with the rest of the device. Figure 26-2 shows the connections between the tile, the CPU interface, and the device.



Copyright © 2024 Texas Instruments Incorporated

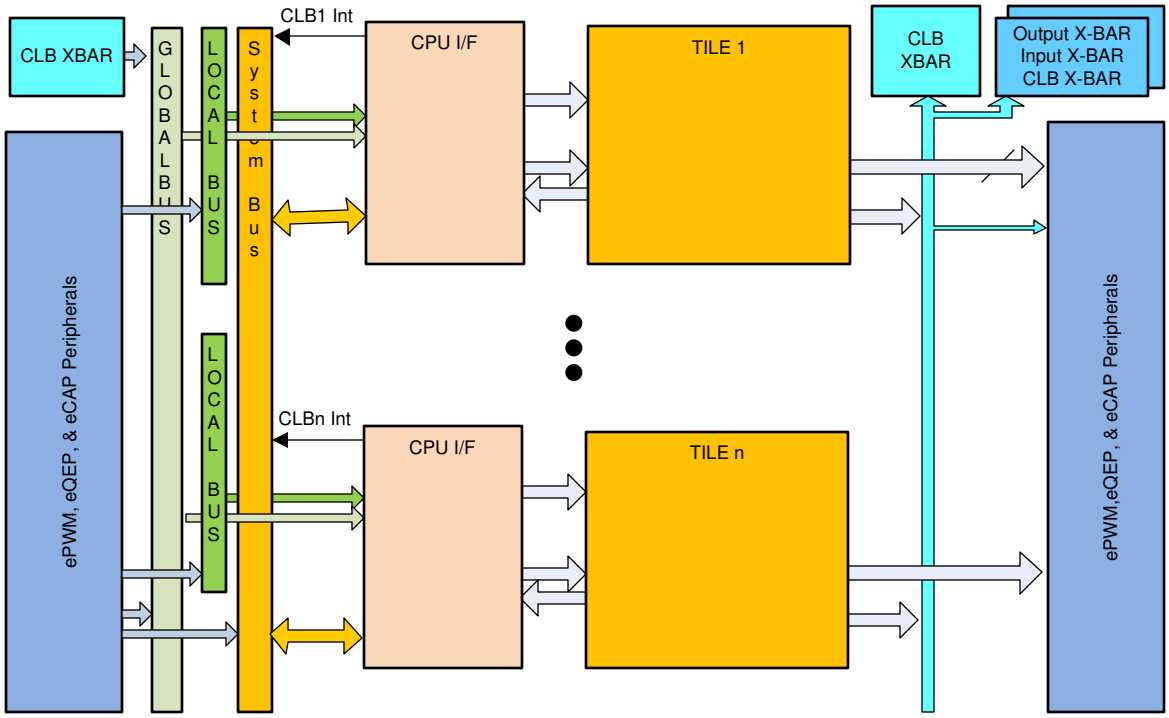



**Figure 26-1. Block Diagram of the CLB Subsystem in the Device**

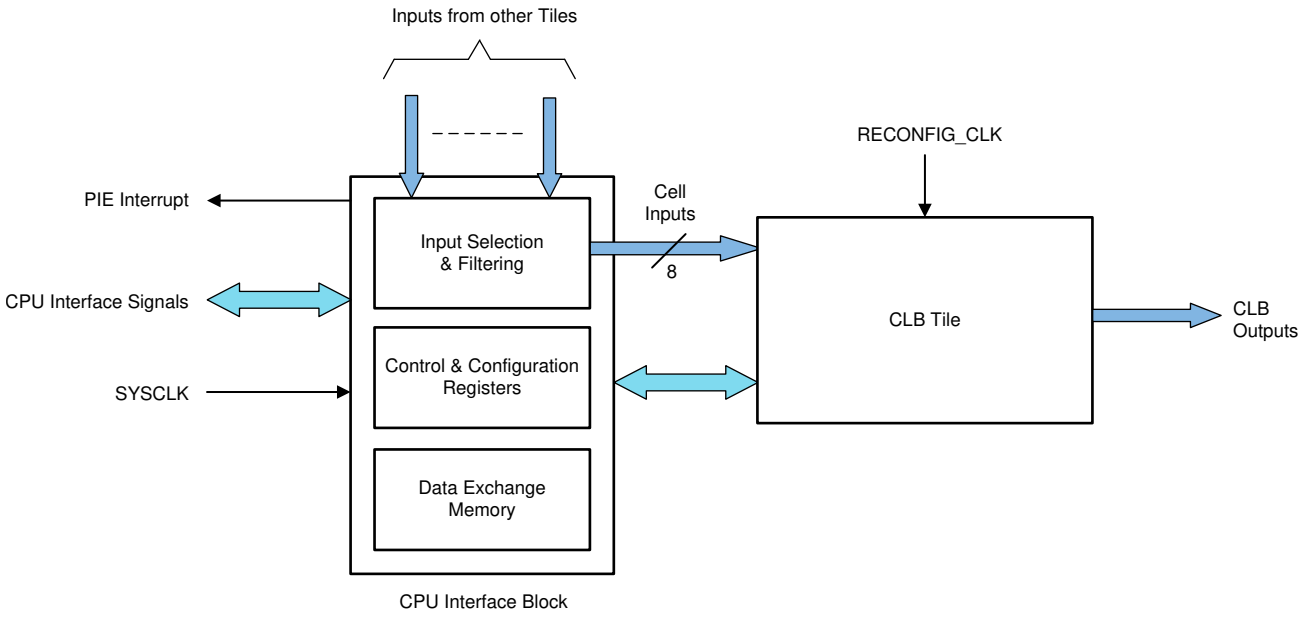



**Figure 26-2. Block Diagram of a CLB Tile and CPU Interface**



Copyright © 2024 Texas Instruments Incorporated

### 26.2.1 CLB Clock

In this device, the CLB clock is the same as the ePWM1 clock, and the maximum frequency is 100MHz. When using CLBx, the corresponding EPWMx module clock bit must be enabled. For example, when CLB1 is being used on the device, EPWM1 clock must be enabled. Likewise, EPWM2 clock must be enabled when making use of CLB2. The other CLBs on the device follow similar requirements.

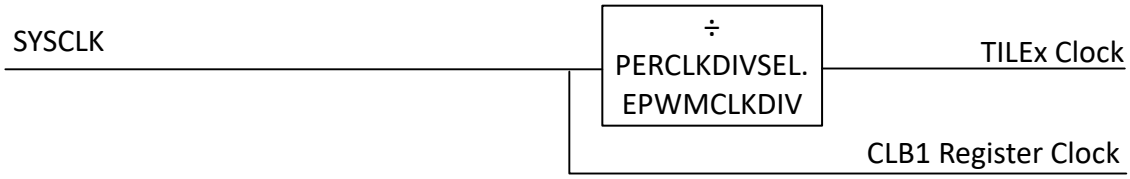



**Figure 26-3. CLB Clocking**



Copyright © 2024 Texas Instruments Incorporated

## 26.3 CLB Input/Output Connection

### *26.3.1 Overview*

There are four instances of the CLB module in the device. Each CLB instance has a common set of input signals referred to as global input signals. Additionally, each CLB instance has a specific set of input signals that are unique to each instance, and are referred to as local input signals. Each of the eight inputs of a CLB can be chosen from any of the global input signals or the local input signals.

**Note**

Signals routed into the CLB using the XBAR must be synchronized within the CLB.

### *26.3.2 CLB Input Selection*

Each CLB module has eight inputs that are applied to the reconfigurable logic cell. Each of these inputs can be selectively driven by a predefined set of signals. A two-level mux structure allows each input of each CLB instance to select a signal.

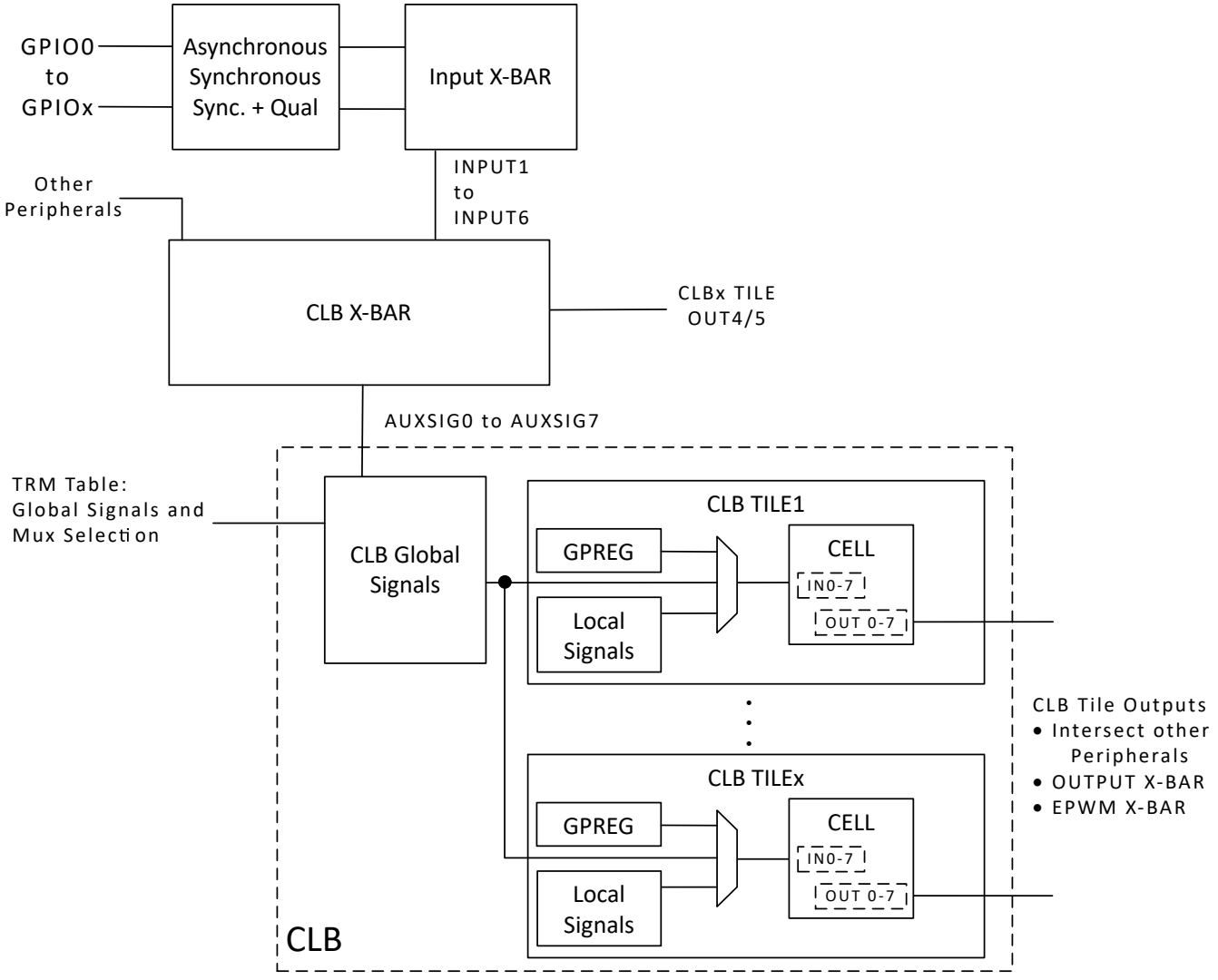



**Figure 26-4. GPIO to CLB Tile Connections**



Copyright © 2024 Texas Instruments Incorporated

A set of signals is common to all the CLB instances. These are referred to as global inputs in Figure 26-5. A separate set of signals is unique to each instance of the CLB. These are referred to as local inputs in Figure 26-5.

Registers CLB_LCL_MUX_SEL_1 and CLB_LCL_MUX_SEL_2 control the local mux selection for each of the eight inputs. The mux control registers CLB_GLBL_MUX_SEL_1 and CLB_GLBL_MUX_SEL_2 control the global mux selection for each of the eight inputs.

The local mux select value of 0 causes the selected global mux input signal to be connected to the corresponding CLB Input. For example, setting CLB_LCL_MUX_SEL_IN_0 = 0 and CLB_GLBL_MUX_SEL_IN_0 = 8 causes the global mux input number 8 to be connected to CLB Input 0. The input filter feature can be used to enable edge detection on the CLB inputs. The input filter feature can also synchronize the input with the CLB clock.

The global mux settings are shown in Table 26-1. The local input mux settings are shown in Table 26-2.

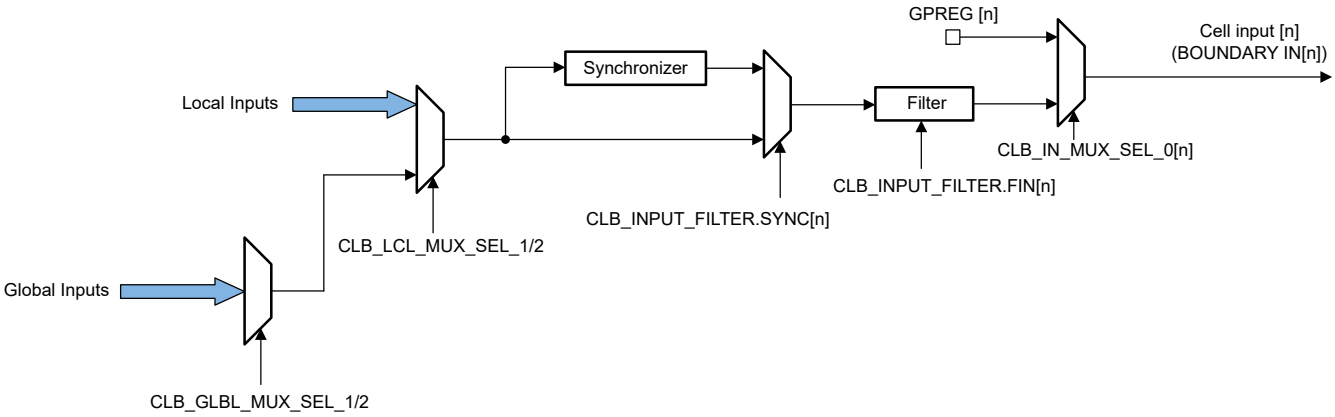



**Figure 26-5. CLB Input Mux and Filter**

Figure 26-6 shows an example of how to use synchronization for an asynchronous signal, in this case the ePWM signal.

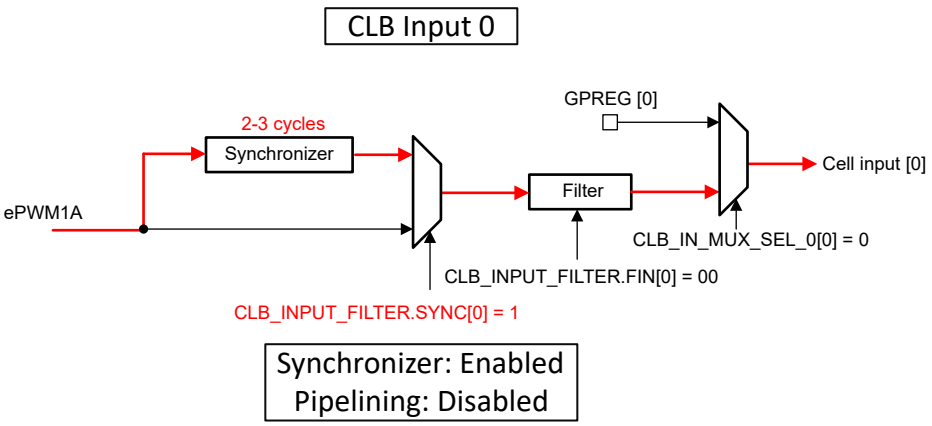



**Figure 26-6. CLB Input Synchronization Example**



Copyright © 2024 Texas Instruments Incorporated

**Note**

If a signal in the following table indicates that synchronization is required, then the CLB input synchronizer must be enabled using the appropriate SYNC bit in the CLB_INPUT_FILTER register. This synchronization adds a 2-3 CLB clock cycle delay to the input. This delay is either 2 or 3 cycles and is not predictable. There is a potential for a metastability hazard, if the indicated signals are not first synchronized before going into the CLB tile. This metastability can cause errors dependent on voltage, temperature, and wafer fab process. Note that this requirement is in addition to and separate from GPIO input synchronization.

**Table 26-1. Global Signals and Mux Selection**

| Select Value | CLB1 Input | CLB2 Input | CLB3 Input | CLB4 Input | Synchronization Requirement |
|---|---|---|---|---|---|
| 0 | EPWM1A | EPWM1A | EPWM1A | EPWM1A | Enable |
| 1 | EPWM1A_OE | EPWM1A_OE | EPWM1A_OE | EPWM1A_OE | Enable |
| 2 | EPWM1B | EPWM1B | EPWM1B | EPWM1B | Enable |
| 3 | EPWM1B_OE | EPWM1B_OE | EPWM1B_OE | EPWM1B_OE | Enable |
| 4 | EPWM1_CTR_ZERO | EPWM1_CTR_ZERO | EPWM1_CTR_ZERO | EPWM1_CTR_ZERO | Disable |
| 5 | EPWM1_CTR_PRD | EPWM1_CTR_PRD | EPWM1_CTR_PRD | EPWM1_CTR_PRD | Disable |
| 6 | EPWM1_CTR_DIR | EPWM1_CTR_DIR | EPWM1_CTR_DIR | EPWM1_CTR_DIR | Disable |
| 7 | EPWM1_TBCLK | EPWM1_TBCLK | EPWM1_TBCLK | EPWM1_TBCLK | Disable |
| 8 | EPWM1_CTR_CMPA | EPWM1_CTR_CMPA | EPWM1_CTR_CMPA | EPWM1_CTR_CMPA | Disable |
| 9 | EPWM1_CTR_CMPB | EPWM1_CTR_CMPB | EPWM1_CTR_CMPB | EPWM1_CTR_CMPB | Disable |
| 10 | EPWM1_CTR_CMPC | EPWM1_CTR_CMPC | EPWM1_CTR_CMPC | EPWM1_CTR_CMPC | Disable |
| 11 | EPWM1_CTR_CMPD | EPWM1_CTR_CMPD | EPWM1_CTR_CMPD | EPWM1_CTR_CMPD | Disable |
| 12 | EPWM1A_AQ | EPWM1A_AQ | EPWM1A_AQ | EPWM1A_AQ | Disable |
| 13 | EPWM1B_AQ | EPWM1B_AQ | EPWM1B_AQ | EPWM1B_AQ | Disable |
| 14 | EPWM1A_DB | EPWM1A_DB | EPWM1A_DB | EPWM1A_DB | Disable |
| 15 | EPWM1B_DB | EPWM1B_DB | EPWM1B_DB | EPWM1B_DB | Disable |
| 16 | EPWM2A | EPWM2A | EPWM2A | EPWM2A | Enable |
| 17 | EPWM2A_OE | EPWM2A_OE | EPWM2A_OE | EPWM2A_OE | Enable |
| 18 | EPWM2B | EPWM2B | EPWM2B | EPWM2B | Enable |
| 19 | EPWM2B_OE | EPWM2B_OE | EPWM2B_OE | EPWM2B_OE | Enable |
| 20 | EPWM2_CTR_ZERO | EPWM2_CTR_ZERO | EPWM2_CTR_ZERO | EPWM2_CTR_ZERO | Disable |
| 21 | EPWM2_CTR_PRD | EPWM2_CTR_PRD | EPWM2_CTR_PRD | EPWM2_CTR_PRD | Disable |
| 22 | EPWM2_CTR_DIR | EPWM2_CTR_DIR | EPWM2_CTR_DIR | EPWM2_CTR_DIR | Disable |
| 23 | EPWM2_TBCLK | EPWM2_TBCLK | EPWM2_TBCLK | EPWM2_TBCLK | Disable |
| 24 | EPWM2_CTR_CMPA | EPWM2_CTR_CMPA | EPWM2_CTR_CMPA | EPWM2_CTR_CMPA | Disable |
| 25 | EPWM2_CTR_CMPB | EPWM2_CTR_CMPB | EPWM2_CTR_CMPB | EPWM2_CTR_CMPB | Disable |
| 26 | EPWM2_CTR_CMPC | EPWM2_CTR_CMPC | EPWM2_CTR_CMPC | EPWM2_CTR_CMPC | Disable |
| 27 | EPWM2_CTR_CMPD | EPWM2_CTR_CMPD | EPWM2_CTR_CMPD | EPWM2_CTR_CMPD | Disable |
| 28 | EPWM2A_AQ | EPWM2A_AQ | EPWM2A_AQ | EPWM2A_AQ | Disable |
| 29 | EPWM2B_AQ | EPWM2B_AQ | EPWM2B_AQ | EPWM2B_AQ | Disable |



Copyright © 2024 Texas Instruments Incorporated

**Table 26-1. Global Signals and Mux Selection (continued)**

| Select Value | CLB1 Input | CLB2 Input | CLB3 Input | CLB4 Input | Synchronization Requirement |
|---|---|---|---|---|---|
| 30 | EPWM2A_DB | EPWM2A_DB | EPWM2A_DB | EPWM2A_DB | Disable |
| 31 | EPWM2B_DB | EPWM2B_DB | EPWM2B_DB | EPWM2B_DB | Disable |
| 32 | EPWM3A | EPWM3A | EPWM3A | EPWM3A | Enable |
| 33 | EPWM3A_OE | EPWM3A_OE | EPWM3A_OE | EPWM3A_OE | Enable |
| 34 | EPWM3B | EPWM3B | EPWM3B | EPWM3B | Enable |
| 35 | EPWM3B_OE | EPWM3B_OE | EPWM3B_OE | EPWM3B_OE | Enable |
| 36 | EPWM3_CTR_ZERO | EPWM3_CTR_ZERO | EPWM3_CTR_ZERO | EPWM3_CTR_ZERO | Disable |
| 37 | EPWM3_CTR_PRD | EPWM3_CTR_PRD | EPWM3_CTR_PRD | EPWM3_CTR_PRD | Disable |
| 38 | EPWM3_CTR_DIR | EPWM3_CTR_DIR | EPWM3_CTR_DIR | EPWM3_CTR_DIR | Disable |
| 39 | EPWM3_TBCLK | EPWM3_TBCLK | EPWM3_TBCLK | EPWM3_TBCLK | Disable |
| 40 | EPWM3_CTR_CMPA | EPWM3_CTR_CMPA | EPWM3_CTR_CMPA | EPWM3_CTR_CMPA | Disable |
| 41 | EPWM3_CTR_CMPB | EPWM3_CTR_CMPB | EPWM3_CTR_CMPB | EPWM3_CTR_CMPB | Disable |
| 42 | EPWM3_CTR_CMPC | EPWM3_CTR_CMPC | EPWM3_CTR_CMPC | EPWM3_CTR_CMPC | Disable |
| 43 | EPWM3_CTR_CMPD | EPWM3_CTR_CMPD | EPWM3_CTR_CMPD | EPWM3_CTR_CMPD | Disable |
| 44 | EPWM3A_AQ | EPWM3A_AQ | EPWM3A_AQ | EPWM3A_AQ | Disable |
| 45 | EPWM3B_AQ | EPWM3B_AQ | EPWM3B_AQ | EPWM3B_AQ | Disable |
| 46 | EPWM3A_DB | EPWM3A_DB | EPWM3A_DB | EPWM3A_DB | Disable |
| 47 | EPWM3B_DB | EPWM3B_DB | EPWM3B_DB | EPWM3B_DB | Disable |
| 48 | EPWM4A | EPWM4A | EPWM4A | EPWM4A | Enable |
| 49 | EPWM4A_OE | EPWM4A_OE | EPWM4A_OE | EPWM4A_OE | Enable |
| 50 | EPWM4B | EPWM4B | EPWM4B | EPWM4B | Enable |
| 51 | EPWM4B_OE | EPWM4B_OE | EPWM4B_OE | EPWM4B_OE | Enable |
| 52 | EPWM4_CTR_ZERO | EPWM4_CTR_ZERO | EPWM4_CTR_ZERO | EPWM4_CTR_ZERO | Disable |
| 53 | EPWM4_CTR_PRD | EPWM4_CTR_PRD | EPWM4_CTR_PRD | EPWM4_CTR_PRD | Disable |
| 54 | EPWM4_CTR_DIR | EPWM4_CTR_DIR | EPWM4_CTR_DIR | EPWM4_CTR_DIR | Disable |
| 55 | EPWM4_TBCLK | EPWM4_TBCLK | EPWM4_TBCLK | EPWM4_TBCLK | Disable |
| 56 | EPWM4_CTR_CMPA | EPWM4_CTR_CMPA | EPWM4_CTR_CMPA | EPWM4_CTR_CMPA | Disable |
| 57 | EPWM4_CTR_CMPB | EPWM4_CTR_CMPB | EPWM4_CTR_CMPB | EPWM4_CTR_CMPB | Disable |
| 58 | EPWM4_CTR_CMPC | EPWM4_CTR_CMPC | EPWM4_CTR_CMPC | EPWM4_CTR_CMPC | Disable |
| 59 | EPWM4_CTR_CMPD | EPWM4_CTR_CMPD | EPWM4_CTR_CMPD | EPWM4_CTR_CMPD | Disable |
| 60 | EPWM4A_AQ | EPWM4A_AQ | EPWM4A_AQ | EPWM4A_AQ | Disable |
| 61 | EPWM4B_AQ | EPWM4B_AQ | EPWM4B_AQ | EPWM4B_AQ | Disable |
| 62 | EPWM4A_DB | EPWM4A_DB | EPWM4A_DB | EPWM4A_DB | Disable |
| 63 | EPWM4B_DB | EPWM4B_DB | EPWM4B_DB | EPWM4B_DB | Disable |
| 64 | AUXSIG0 | AUXSIG0 | AUXSIG0 | AUXSIG0 | Enable |
| 65 | AUXSIG1 | AUXSIG1 | AUXSIG1 | AUXSIG1 | Enable |
| 66 | AUXSIG2 | AUXSIG2 | AUXSIG2 | AUXSIG2 | Enable |



Copyright © 2024 Texas Instruments Incorporated

**Table 26-1. Global Signals and Mux Selection (continued)**

| Select Value | CLB1 Input | CLB2 Input | CLB3 Input | CLB4 Input | Synchronization Requirement |
|---|---|---|---|---|---|
| 67 | AUXSIG3 | AUXSIG3 | AUXSIG3 | AUXSIG3 | Enable |
| 68 | AUXSIG4 | AUXSIG4 | AUXSIG4 | AUXSIG4 | Enable |
| 69 | AUXSIG5 | AUXSIG5 | AUXSIG5 | AUXSIG5 | Enable |
| 70 | AUXSIG6 | AUXSIG6 | AUXSIG6 | AUXSIG6 | Enable |
| 71 | AUXSIG7 | AUXSIG7 | AUXSIG7 | AUXSIG7 | Enable |
| 72-127 | Reserved | Reserved | Reserved | Reserved | Reserved |

**Note**

EPWMxA_OE and EPWMxB_OE refer to trip outputs from the respective EPWM module.

EPWMxA_AQ and EPWMxB_AQ refer to the output of the AQ submodule in the respective EPWM module.

EPWMxA_DB and EPWMBx_DB refer to the output of the DB submodule in the respective EPWM module.

**Note**

If a signal in the following table indicates that synchronization is required, then the CLB input synchronizer must be enabled using the appropriate SYNC bit in the CLB_INPUT_FILTER register. This synchronization adds a 2-3 CLB clock cycle delay to the input. This delay is either 2 or 3 cycles and is not predictable. There is a potential for a metastability hazard, if the indicated signals are not first synchronized before going into the CLB tile. This metastability can cause errors dependent on voltage, temperature, and wafer fab process. Note that this requirement is in addition to and separate from GPIO input synchronization.

**Table 26-2. Local Signals and Mux Selection**

| Select Value | CLB1 Input | CLB2 Input | CLB3 Input | CLB4 Input | Synchronization Requirement |
|---|---|---|---|---|---|
| 0 | CLB1_GLB_MUX_OUT | CLB2_GLB_MUX_OUT | CLB3_GLB_MUX_OUT | CLB4_GLB_MUX_OUT | Enable |
| 1 | EPWM1_DCAEVT1 | EPWM2_DCAEVT1 | EPWM3_DCAEVT1 | EPWM4_DCAEVT1 | Enable |
| 2 | EPWM1_DCAEVT2 | EPWM2_DCAEVT2 | EPWM3_DCAEVT2 | EPWM4_DCAEVT2 | Enable |
| 3 | EPWM1_DCBEVT1 | EPWM2_DCBEVT1 | EPWM3_DCBEVT1 | EPWM4_DCBEVT1 | Enable |
| 4 | EPWM1_DCBEVT2 | EPWM2_DCBEVT2 | EPWM3_DCBEVT2 | EPWM4_DCBEVT2 | Enable |
| 5 | EPWM1_DCAH | EPWM2_DCAH | EPWM3_DCAH | EPWM4_DCAH | Enable |
| 6 | EPWM1_DCAL | EPWM2_DCAL | EPWM3_DCAL | EPWM4_DCAL | Enable |
| 7 | EPWM1_DCBH | EPWM2_DCBH | EPWM3_DCBH | EPWM4_DCBH | Enable |
| 8 | EPWM1_DCBL | EPWM2_DCBL | EPWM3_DCBL | EPWM4_DCBL | Enable |
| 9 | EPWM1_OST | EPWM2_OST | EPWM3_OST | EPWM4_OST | Enable |
| 10 | EPWM1_CBC | EPWM2_CBC | EPWM3_CBC | EPWM4_CBC | Enable |
| 11 | ECAP1IN0 | ECAP2IN0 | ECAP3IN0 | ECAP4IN0 | Enable |
| 12 | ECAP1_OUT | ECAP2_OUT | ECAP3_OUT | ECAP4_OUT | Disable |
| 13 | ECAP1_OUT_EN | ECAP2_OUT_EN | ECAP3_OUT_EN | ECAP4_OUT_EN | Disable |



Copyright © 2024 Texas Instruments Incorporated

**Table 26-2. Local Signals and Mux Selection (continued)**

| Select Value | CLB1 Input | CLB2 Input | CLB3 Input | CLB4 Input | Synchronization Requirement |
|---|---|---|---|---|---|
| 14 | ECAP1_CEVT1 | ECAP2_CEVT1 | ECAP3_CEVT1 | ECAP4_CEVT1 | Disable |
| 15 | ECAP1_CEVT2 | ECAP2_CEVT2 | ECAP3_CEVT2 | ECAP4_CEVT2 | Disable |
| 16 | ECAP1_CEVT3 | ECAP2_CEVT3 | ECAP3_CEVT3 | ECAP4_CEVT3 | Disable |
| 17 | ECAP1_CEVT4 | ECAP2_CEVT4 | ECAP3_CEVT4 | ECAP4_CEVT4 | Disable |
| 18 | EQEP1A | EQEP2A | EQEP3A | Reserved | Enable |
| 19 | EQEP1B | EQEP2B | EQEP3B | Reserved | Enable |
| 20 | EQEP1I | EQEP2I | EQEP3I | Reserved | Enable |
| 21 | EQEP1S | EQEP2S | EQEP3S | Reserved | Enable |
| 22 | CPU1_TBCLKSYNC | CPU1_TBCLKSYNC | Reserved | Reserved | Disable |
| 23 | CPU2_TBCLKSYNC | CPU2_TBCLKSYNC | Reserved | Reserved | Disable |
| 24 | CPU1_HALT | CPU1_HALT | Reserved | Reserved | Disable |
| 25 | CPU2_HALT | CPU2_HALT | Reserved | Reserved | Disable |
| 26-31 | Reserved | Reserved | Reserved | Reserved | Reserved |

The GPREG is accessible by the CPU and the bits of this register can be used as BOUNDARY INPUTs for the CLB Tiles. For example, CLB1s GPREG[0] can be used as BOUNDARY IN0 (Cell Input 0) for the corresponding CLB Tile.

To connect multiple tiles to each other, you can use the CLBx OUT4/5 and connect to CLBy BOUNDARY INz through the CLB X-BAR and the Global Signals Mux.

Another option is to connect the CLBx OUT0-7 to a GPIO and then use the INPUT X-BAR to bring the signal back in to the device and connect to the CLBy BOUNDARY INz through the CLB X-BAR and the Global Signals Mux.

To use GPIOs as inputs to the CLB, you must utilize the Input X-BAR and the CLB X-BAR. Figure 26-4 shows how GPIOs can be used as inputs to the CLB tiles.



Copyright © 2024 Texas Instruments Incorporated

### 26.3.3 CLB Output Selection

The eight outputs of the CLB are replicated to create 16 output signals. Each of these 16 outputs has a separate enable bit defined in the CLB output enable register, CLB_OUT_EN. The CLB outputs go to the ePWM, eCAP, eQEP and the crossbar module in the device. This allows the user to enhance the functionality of these modules with the CLB. Figure 26-7 shows the CLB outputs.

**Note**

The output from OUTLUT0 is connected to OUT0 and OUT8. While the signal is the same, each OUTy has access to a different peripheral as shown in Section 26.3.4. Likewise, OUTLUT1 is connected to OUT1 and OUT9, and are the same signal.

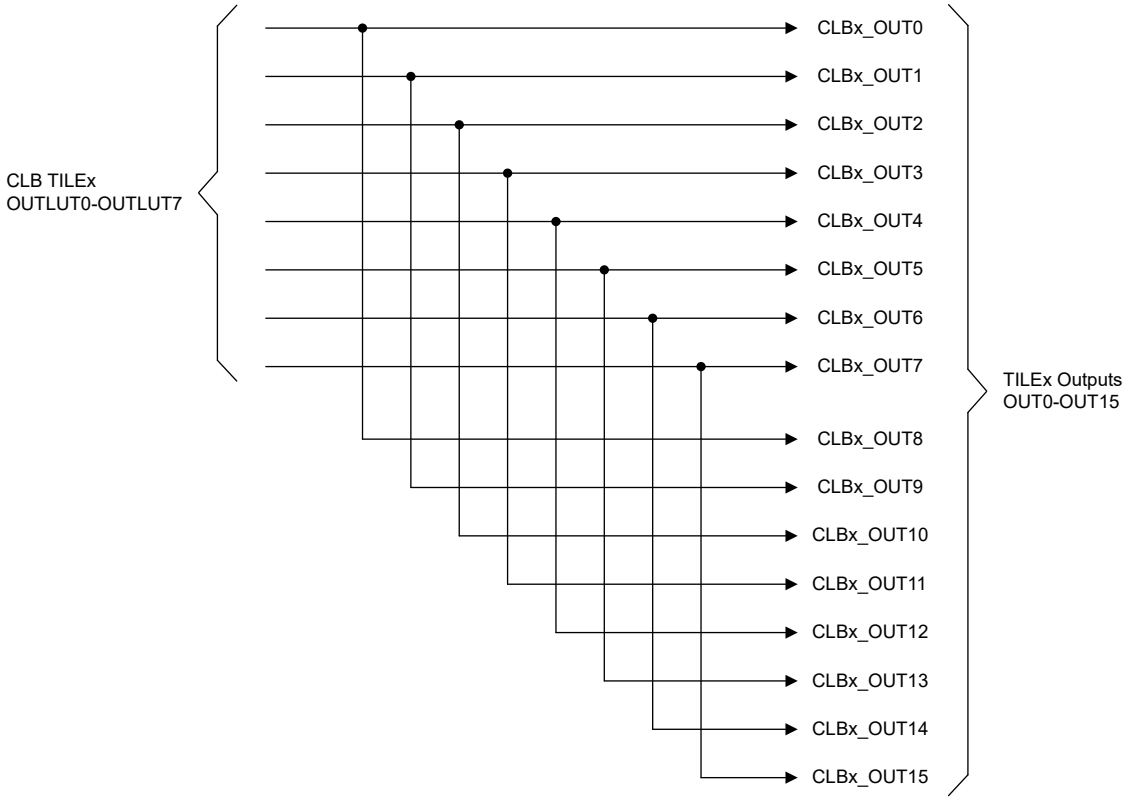



**Figure 26-7. CLB Outputs**



Copyright © 2024 Texas Instruments Incorporated

### 26.3.4 CLB Output Signal Multiplexer

Each CLB output signal passes through an external multiplexer that intersects a specific peripheral signal, see Figure 26-8. The output of the multiplexer is connected to the destination of the original peripheral signal and the default multiplexer setting is that the peripheral signal is passed through. The multiplexer is controlled by bit[n] in the CLB output enable register CLB_OUT_EN.

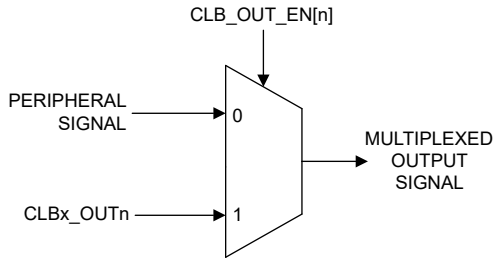



**Figure 26-8. CLB Output Signal Multiplexer**

For example, if the CLB1 OUT0 must override the EPWM1A signal, the OUPTUT ENABLE bit for OUT0 must be set to 1.

Table 26-3 shows the allocation of peripheral signals and the CLB outputs.

**Table 26-3. CLB Output Signal Multiplexer Table**

| CLB Output | CLB OUTLUT | CLB1 Destination | CLB2 Destination | CLB3 Destination | CLB4 Destination |
|---|---|---|---|---|---|
| 0 | OUTLUT0 | EPWM1A | EPWM2A | EPWM3A | EPWM4A |
| 1 | OUTLUT1 | EPWM1A_OE | EPWM2A_OE | EPWM3A_OE | EPWM4A_OE |
| 2 | OUTLUT2 | EPWM1B | EPWM2B | EPWM3B | EPWM4B |
| 3 | OUTLUT3 | EPWM1B_OE | EPWM2B_OE | EPWM3B_OE | EPWM4B_OE |
| 4 | OUTLUT4 | EPWM1A_AQ | EPWM2A_AQ | EPWM3A_AQ | EPWM4A_AQ |
| 5 | OUTLUT5 | EPWM1B_AQ | EPWM2B_AQ | EPWM3B_AQ | EPWM4B_AQ |
| 6 | OUTLUT6 | EPWM1A_DB | EPWM2A_DB | EPWM3A_DB | EPWM4A_DB |
| 7 | OUTLUT7 | EPWM1B_DB | EPWM2B_DB | EPWM3B_DB | EPWM4B_DB |
| 8 | OUTLUT0 | EQEP1_QCLK | EQEP2_QCLK | EQEP3_QCLK | Reserved |
| 9 | OUTLUT1 | EQEP1_QDIR | EQEP2_QDIR | EQEP3_QDIR | Reserved |
| 10 | OUTLUT2 | Reserved | Reserved | Reserved | Reserved |
| 11 | OUTLUT3 | Reserved | Reserved | Reserved | Reserved |
| 12 | OUTLUT4 | XBARs | XBARs | XBARs | XBARs |
| 13 | OUTLUT5 | XBARs | XBARs | XBARs | XBARs |
| 14 | OUTLUT6 | ECAP1_OUT | ECAP2_OUT | ECAP3_OUT | ECAP4_OUT |
| 15 | OUTLUT7 | ECAP1_OUT_EN | ECAP2_OUT_EN | ECAP3_OUT_EN | ECAP4_OUT_EN |

**Note**

When not explicitly specified in the table above, refer to corresponding IP chapters for exact CLB Output destinations (that is, FSI, ECAP, XBAR connections).



Copyright © 2024 Texas Instruments Incorporated

## 26.4 CLB Tile

The purpose of the CLB tile is to provide the logic reconfiguration capability of the CLB. The CLB tile contains the following submodules:

- **Counter:** The counter submodule can be configured as an adder, a counter, or a shifter. When functioning as an adder, the counter submodule can either add or subtract. When functioning as a counter, the counter submodule can count up or count down. When functioning as a shifter, the counter submodule can shift left or shift right. The counter event inputs, as well as the reset input, can be freely connected to any of the other submodules in the same tile. There are three counters in each tile.
- **LUT4:** The LUT4 submodule has a 4-input look-up table functionality and is capable of realizing any combinatorial Boolean equation of up to four inputs. There are three LUT4 submodules in each CLB tile.
- **FSM:** The Finite State Machine (FSM) submodule can be configured either as a single four-state finite state machine, or as two independent two-state finite state machines. The FSM accepts two external inputs, and generates two state outputs and one combination output. When not used as a state machine, the FSM submodule can accept two external inputs and function as a 4-input LUT. There are three FSM submodules in each CLB tile.
- **Output LUT:** The output LUT is a 3-input lookup table submodule capable of realizing any combinatorial Boolean equation of up to three inputs. There are eight such blocks in a CLB tile, each associated with one of the tile outputs.
- **High Level Controller:** The High Level Controller (HLC) submodule is an event-driven block that can handle up to four concurrent events. The event can be an activity on any of the other block outputs. A predefined set of operations is executed when each event occurs. The HLC also provides a data exchange and interrupt mechanism to the CPU subsystem. There are four working registers (R0, R1, R2, and R3) that can be used for basic operations, and to modify or set up values for the three counter blocks. Unlike the other submodules, there is only one HLC in each CLB tile.
- **Static Switch Block:** The static switch block provides dynamic connectivity between all of the blocks listed above. Submodules can be connected by the user, with the only restriction that the submodules must not form a combinational loop within the tile.

A CLB tile consists of three sets each of the counter block, FSM, and LUT4, one high-level controller, and eight output LUT blocks. The submodule numbering is shown in Figure 26-9.

The functionality of the LUT submodules is configured using a register field containing the binary pattern of the output of the desired look-up table. For example, a 4-input LUT has 16 possible input permutations, each of which corresponds to a desired binary 0 or 1 at the output. The register field can, therefore, be 16-bits in length, with each bit representing the desired result of a binary pattern. Input pattern sequences start at 0000 and continue sequentially to 1111. A similar method is used to encode the 16-bit state equations in the FSM submodule.

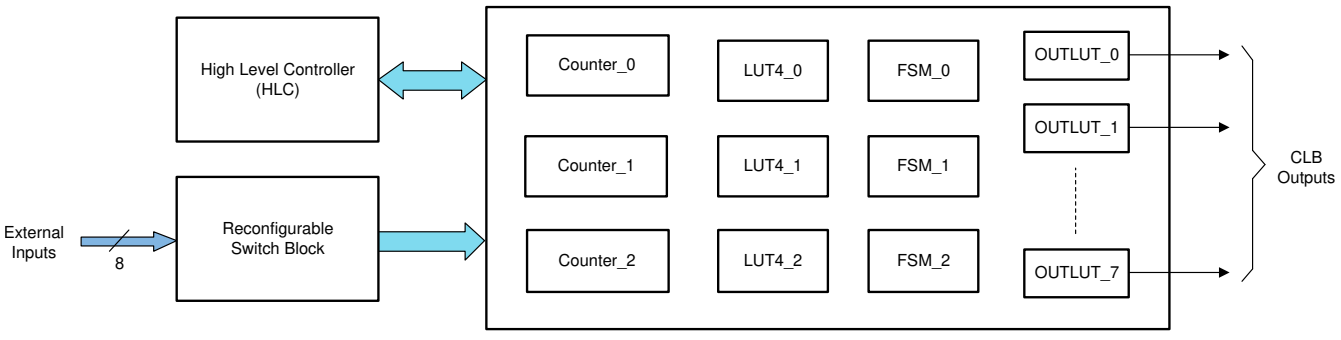



**Figure 26-9. CLB Tile Submodules**



Copyright © 2024 Texas Instruments Incorporated

### 26.4.1 Static Switch Block

The Static Switch Block provides the configurable connectivity between the submodules in the CLB tile. The outputs of all the submodules and the eight external inputs are connected to a common internal bus inside the tile. Every input port has a 32-to-1 multiplexer and an associated 5-bit selection value that allows the user to select one of the inputs on the bus. The only restrictions are certain signals (described below) that are tied off in the design to prevent creation of accidental combinatorial loops.

**Table 26-4. Output Table**

| Bit Position | Signal Connection | Comment |
|---|---|---|
| 0 | Always 0 | This select value is used to tie an input to 0. |
| 1 | COUNTER_0 MATCH2 | |
| 2 | COUNTER_0 ZERO | |
| 3 | COUNTER_0 MATCH1 | |
| 4 | FSM_0 STATE_BIT_0 | |
| 5 | FSM_0 STATE_BIT_1 | |
| 6 | FSM_0 LUT output | |
| 7 | LUT4_0 output | |
| 8 | Always 1 | This select value is used to tie an input to 1. |
| 9 | COUNTER_1 MATCH2 | |
| 10 | COUNTER_1 ZERO | |
| 11 | COUNTER_1 MATCH1 | |
| 12 | FSM_1 STATE_BIT_0 | |
| 13 | FSM_1 STATE_BIT_1 | |
| 14 | FSM_1 LUT output | |
| 15 | LUT4_1 output | |
| 16 | Always ‘0’ | |
| 17 | COUNTER_2 MATCH2 | |
| 18 | COUNTER_2 ZERO | |
| 19 | COUNTER_2 MATCH1 | |
| 20 | FSM_2 STATE_BIT_0 | |
| 21 | FSM_2 STATE_BIT_1 | |
| 22 | FSM_2 LUT output | |
| 23 | LUT4_2 output | |
| 24 | External Input 0 | |
| 25 | External Input 1 | |
| 26 | External Input 2 | |
| 27 | External Input 3 | |
| 28 | External Input 4 | |
| 29 | External Input 5 | |
| 30 | External Input 6 | |
| 31 | External Input 7 | |



Copyright © 2024 Texas Instruments Incorporated

**Table 26-5. Input Table**

| Module Name | Port Name | Description |
|---|---|---|
| Counter Block | RESET | Acts as an active high reset when used as a counter |
| | MODE_0 | Acts as an enable when used as a counter. The counter counts only when this input is 1. |
| | MODE_1 | Acts as a direction control when used as a counter. If this input is 1, then the counter counts up; else, the counter counts down. |
| LUT | IN0 | Input 0 of the 4-input LUT. |
| | IN1 | Input 1 of the 4-input LUT. |
| | IN2 | Input 2 of the 4-input LUT. |
| | IN3 | Input 3 of the 4-input LUT. |
| FSM | EXT_IN0 | Input 0 of the FSM block. |
| | EXT_IN1 | Input 1 of the FSM block. |
| | EXTRA_EXT_IN0 | Extra external input 0 of the FSM block. This input matters only if configured in the LUT mode. |
| | EXTRA_EXT_IN1 | Extra external input 1 of the FSM block. This input matters only if configured in the LUT mode. |

The static switch block allows the user to define the input connection of any submodule to come from any of the outputs in Table 26-4. It is therefore easy to create a combinatorial loop. To prevent this, certain paths are broken in the input path of each submodule. These port positions are tied to 0, as shown in Table 26-6.

**Table 26-6. Ports Tied Off to Prevent Combinatorial Loops**

| Module Name | Ports of Input MUX Tied Off to 0 to Prevent Combinational Loops |
|---|---|
| LUT_0 | LUT_0 , LUT_1, and LUT_2 output, FSM_0, FSM_1, and FSM_2 output. |
| FSM_0 | LUT_1 and LUT_2 output, FSM_0, FSM_1, and FSM_2 output. |
| LUT_1 | LUT_1 and LUT_2 output, FSM_1 and FSM_2 output. |
| FSM_1 | LUT_2 output, FSM_1 and FSM_2 output. |
| LUT_2 | LUT_2 output, FSM_2 output. |
| FSM_2 | FSM_2 output. |



Copyright © 2024 Texas Instruments Incorporated

### 26.4.2 Counter Block

#### 26.4.2.1 Counter Description

The counter block is a complex functional submodule that can be configured either as a counter, an adder, or a shifter. Apart from the normal operational control, this block has a dedicated EVENT input, which can trigger an addition, subtraction or shift operation, or load data into the counter register. The inputs to the counter submodule are shown in Figure 26-10.

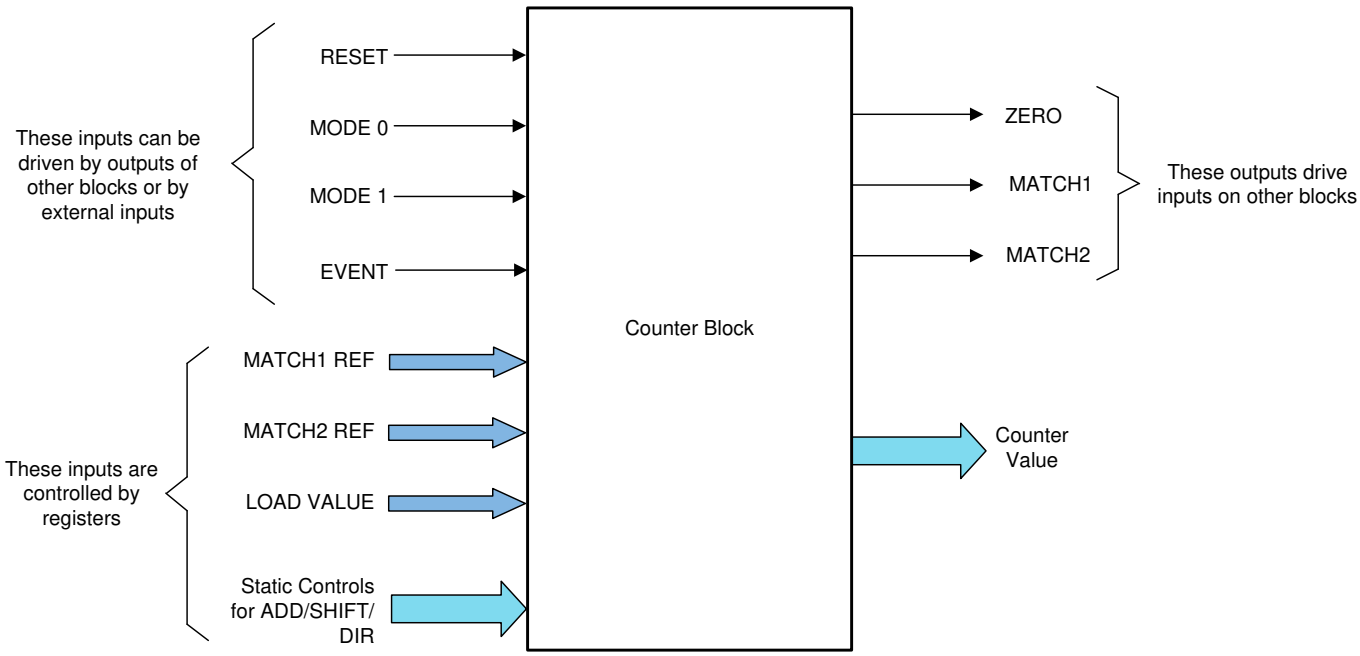



**Figure 26-10. Counter Block**



Copyright © 2024 Texas Instruments Incorporated

#### 26.4.2.2 Counter Operation

At the heart of the counter block is a 32-bit count register. This register can either be loaded statically before counting commences, or dynamically at run time. The operation of the counter submodule is determined by the inputs described below. Note that each of the inputs can be connected to the outputs of any of the other blocks in the CLB tile. These connections are made by configuring the static switch block.

The counter inputs are:

- **RESET:** This is the highest priority input and takes precedence over all other inputs. The input is level sensitive and as long as the input remains high, the counter resets to 0 on the next clock cycle.
- **MODE_0:** This input is an enable for the counter. The counter begins counting (up or down depending on the MODE_1 setting) only when this input is high. If this input is low, then no counting takes place.
- **MODE_1:** This input is the direction control for the counter. If this input is high, then the counter increments on every clock cycle where MODE_0 is high. If this input is low, then the counter decrements on every clock cycle where MODE_0 is high. The counter wraps around to 0x0000 0000 after 0xFFFF FFFF when counting up. The counter wraps around to 0xFFFF FFFF after 0x0000 0000 when counting down. The only exception to this is when an EVENT occurs at exactly the same time, causing a different value to be loaded into the counter.
- **EVENT:** This input is defined for the purpose of triggering actions in the counter based on certain events. The event can be any of the outputs of the other blocks or an external input to the tile. The counter's static control inputs define the behavior of the counter on an active event. An active event is defined as a rising edge on the EVENT input. The counter can be configured to perform one of the following actions:
  - Load a predefined 32-bit value from the LOAD VALUE register into the count register
  - Shift the contents of the counter register left or right by a predefined amount between 0 and 31
  - Add or subtract a predefined 32-bit value. Addition and subtraction are treated as 32-bit unsigned operations and there is no saturation.

  Note that the effect of a rising edge on the EVENT input only lasts for one cycle. On the next cycle, the counter operation continues based on the MODE_0, MODE_1, and RESET inputs.

MATCH1 REF and MATCH2 REF are 32-bit reference values that are used to generate the MATCH1 and MATCH2 outputs. The MATCH1 output becomes active high whenever the counter register value matches the 32-bit MATCH1 REF value. MATCH2 behaves in a similar manner in relation to the MATCH2 REF register. The reference values for MATCH1 and MATCH2 can either be setup once before the start of operation, or can be modified dynamically. The High Level Controller can load desired values into the MATCH1 REF and MATCH2 REF registers.

Note that the counter load and match registers are not memory-mapped. For more information, see Section 26.5.2.

The three logic outputs of the counter block are:

- **ZERO:** This output goes high whenever the counter register is 0.
- **MATCH1:** This output goes high whenever the counter register is equal to the MATCH1 REF input register.
- **MATCH2:** This output goes high whenever the counter register is equal to the MATCH2 REF input register.

The operation of the counter block is controlled by the CFG_MISC_CTRL register. The following three bits of this register are relevant for each counter. The "x" below refers to the counter instance; 0, 1, or 2. For more information, see the CLB_MISC_CONTROL register description located in Section 26.8.

- COUNT_EVENT_CTRL_x: This bit defines whether the counter performs an addition or a shift on an event. A value of 0 means that on an event, the counter loads the static value; 1 means an add/shift operation is performed. This bit must be 0 for indirect loads and HLC loads of the counter to take effect.
- COUNT_ADD_SHIFT_x. 1 means add, 0 means shift.
- COUNT_DIR_x. 1 means left shift or add. 0 means right shift or subtract.



Copyright © 2024 Texas Instruments Incorporated

Table 26-7 shows the logical operation of the counter block in terms of the inputs and control register bits. Count up and down modes are the normal operation with EVENT = 0. The operations on the CNTVAL register are:

Load: CNTVAL = EVENT_LOAD_VAL

Shift right: CNTVAL = CNTVAL >> EVENT_LOAD_VAL

Shift left: CNTVAL = CNTVAL << EVENT_LOAD_VAL

Subtract: CNTVAL = CNTVAL - EVENT_LOAD_VAL

Add: CNTVAL = CNTVAL + EVENT_LOAD_VAL

**Table 26-7. Counter Block Operating Modes**

| EVENT | MODE_0 | MODE_1 | COUNT_EVENT_CTRL_x | COUNT_ADD_SHIFT_x | COUNT_DIR_x | Action on CNTVAL |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | X | X | X | None |
| 0 | 0 | 1 | X | X | X | None |
| 0 | 1 | 0 | X | X | X | Count down |
| 0 | 1 | 1 | X | X | X | Count up |
| 1 | X | X | 0 | X | X | Load |
| 1 | X | X | 1 | 0 | 0 | Shift right |
| 1 | X | X | 1 | 0 | 1 | Shift left |
| 1 | X | X | 1 | 1 | 0 | Subtract |
| 1 | X | X | 1 | 1 | 1 | Add |



Copyright © 2024 Texas Instruments Incorporated

### 26.4.3 FSM Block

The Finite State Machine (FSM) block provides the ability to build programmable finite state machines with up to four states. The FSM block has two register bits and two external inputs, and can be programmed either as two 2-state machines or as a single 4-state machine. For additional flexibility, there are two auxiliary inputs (EXTRA_EXT_IN0 and EXTRA_EXT_IN1) that can be used to create larger combinational functions by giving up a state functionality. The structure of the FSM is shown in Figure 26-11.

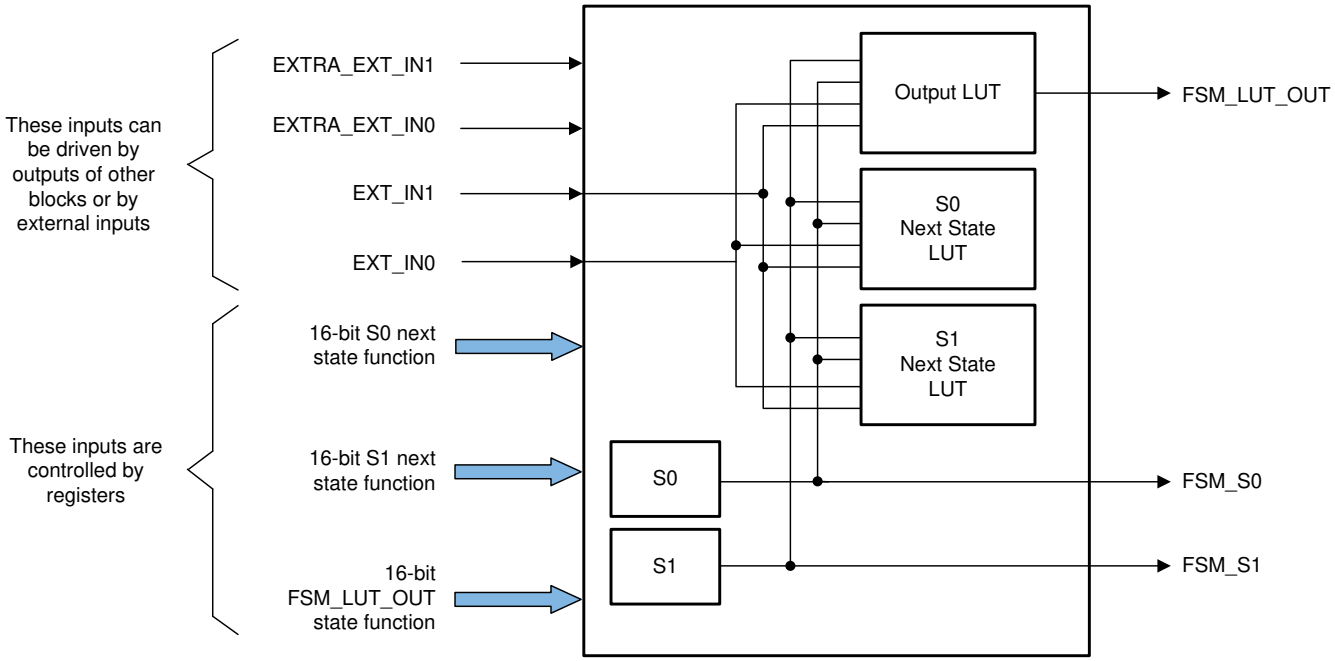



**Figure 26-11. FSM Block**



Copyright © 2024 Texas Instruments Incorporated

The signals and functionality of the FSM block are:

- **EXT_IN0 and EXT_IN1:** These are the two external inputs that can be used to control the output FSM_LUT_OUT or either of the states S0 and S1.
- **S0** and **S1** are two state bits that have independent state control equations.
- **16-bit S0 equation** defines a function (EXT_IN1, EXT_IN0, S1, S0). The four bits in the order defined are used as an index into the 16-bit register to decide the next state of S0.
- **16-bit S1 equation** defines a function (EXT_IN1, EXT_IN0, S1, S0). The four bits in the order defined are used as an index into the 16-bit register to decide the next state of S1.
- **16-bit output equation** defines a function (EXT_IN1, EXT_IN0, S1, S0). The four bits in the order defined are used as an index into the 16-bit register to decide the output value of FSM_LUT_OUT. An additional level of configurability is provided such that FSM_LUT_OUT can use extra inputs in case the states S0 and S1 are unused.

  One extra bit is used to select EXTRA_EXT_IN0 instead of S0. One extra bit is used to select EXTRA_EXT_IN1 instead of S1. Using these, one can effectively build 3-input or a 4-input LUT for the FSM_LUT_OUT by giving up one or two state bits, respectively.

The CFG_MISC_CTRL register controls the operation of the FSM block. Two bits in this register are used for each FSM Block to determine whether the FSM output LUT function uses the state variable S0/S1, or the corresponding extra external input signal FSM_EXTRA_EXT_INx. A 0 means use the state bit, and a 1 means use the FSM_EXTRA_EXT_IN0 / FSM_EXTRA_EXT_IN1 signal.

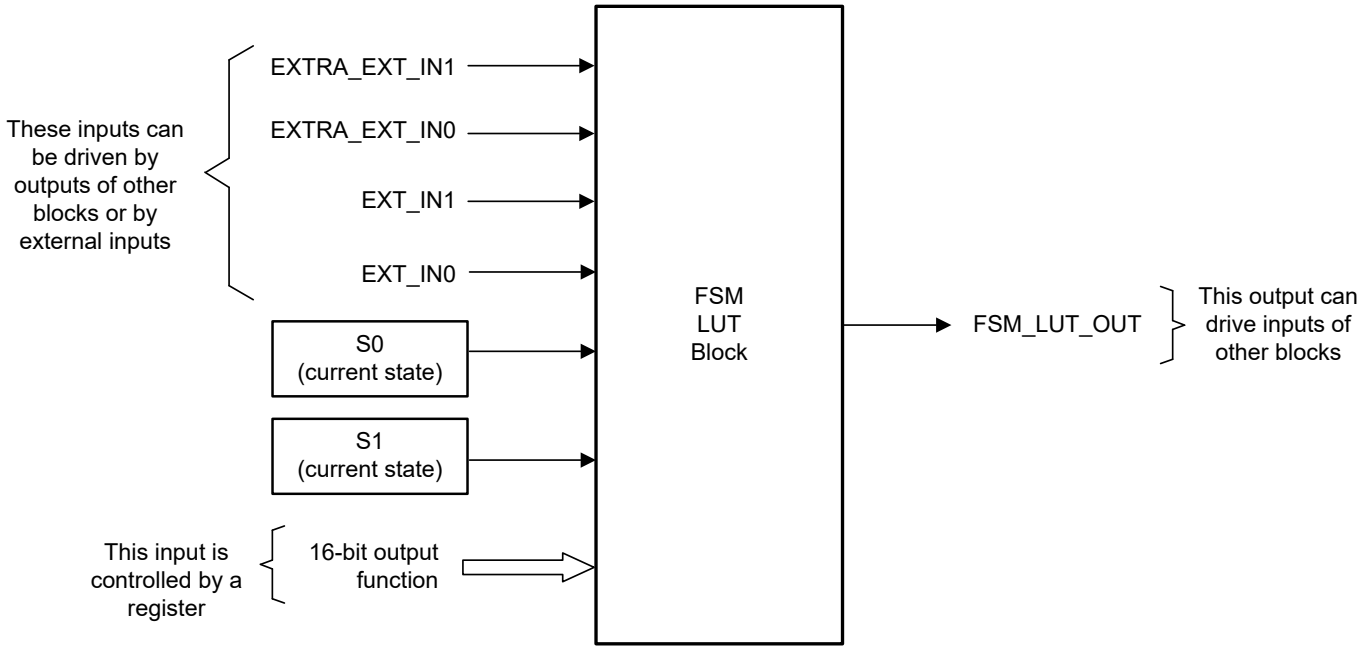



**Figure 26-12. FSM LUT Block**



Copyright © 2024 Texas Instruments Incorporated

### 26.4.4 LUT4 Block

This is a simple four input Look-Up table (LUT) block with inputs IN0, IN1, IN2, and IN3 (see Figure 26-13). Any combinatorial Boolean equation using the four inputs can be realized by programming the 16-bit control register associated with each LUT4 block. For more information, see the LUT4 register descriptions located in Section 26.8.

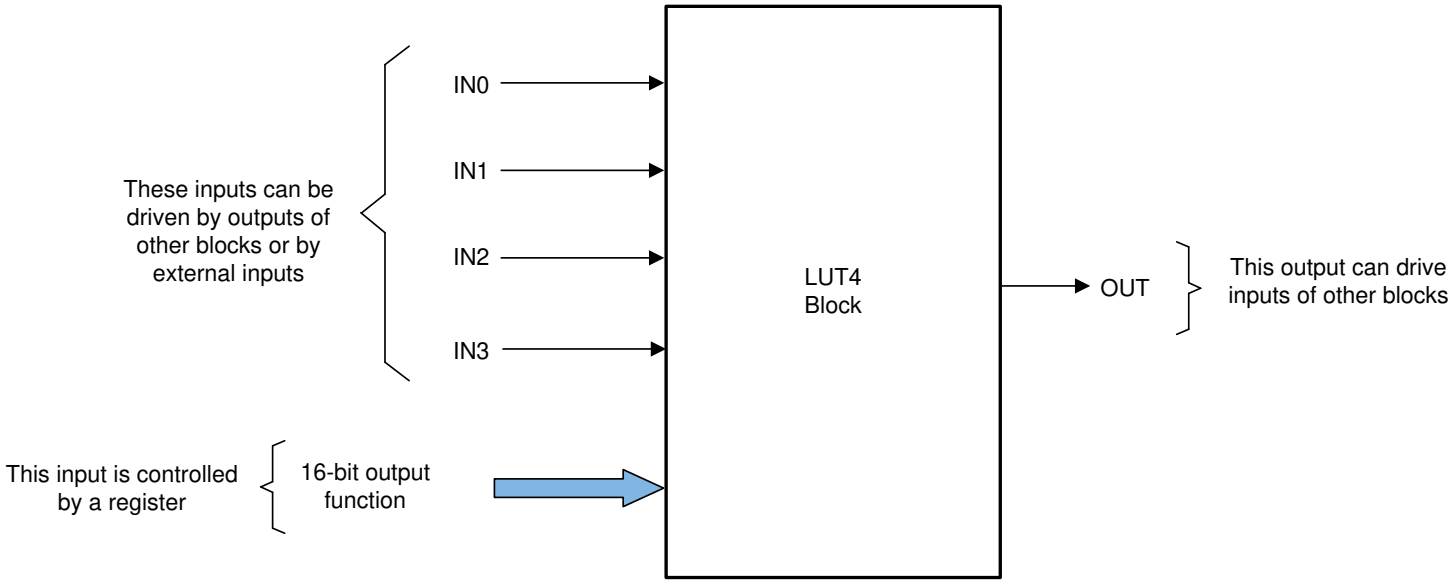



**Figure 26-13. LUT4 Block**

### 26.4.5 Output LUT Block

The output LUT block (Figure 26-14) is very similar in functionality to the LUT4 block, except that the output LUT block has three inputs. Unlike the other sub blocks, the outputs of these blocks are meant to go out of the tile and hence cannot be used by any other block within the tile. Any combinatorial function of the three inputs can be realized by the output LUT block. For more information, see the output LUT register descriptions located in Section 26.8.

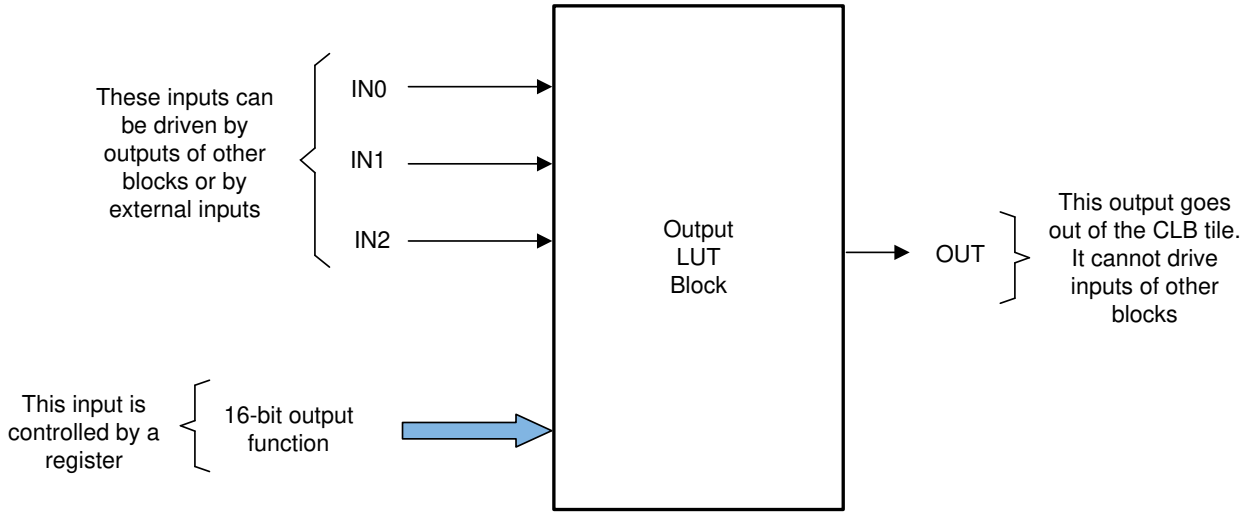



**Figure 26-14. Output LUT Block**



Copyright © 2024 Texas Instruments Incorporated

### 26.4.6 High Level Controller (HLC)

The High Level Controller (HLC) is significantly more complex than the other blocks in the CLB tile. The HLC performs two main functions:

- Provides a means of communication and data exchange with the rest of the device. This is done through two methods: a global access path to four general purpose HLC registers (R0 through R3) and PUSH and PULL FIFOs between the CPU and the HLC. The general-purpose HLC registers are designed to only be written to during device configuration time. To avoid unexpected behavior, the HLC registers must not be written to during run-time operation. The PUSH and PULL FIFOs are the primary avenues of data exchange during run-time, refer to Section 26.4.6.4 for more information.
- Provides a programmable, event-based action system, which performs computation, manipulation of logic functionality, and data movement. In other words, events can be configured to trigger a predefined set of actions in the CLB tile, or to initiate data exchange with the rest of the device.

The architecture of the HLC is shown in Figure 26-15. The HLC is an event-based system capable of handling up to four simultaneous events that can be selected from any outputs of the other blocks within the tile or from an external input.

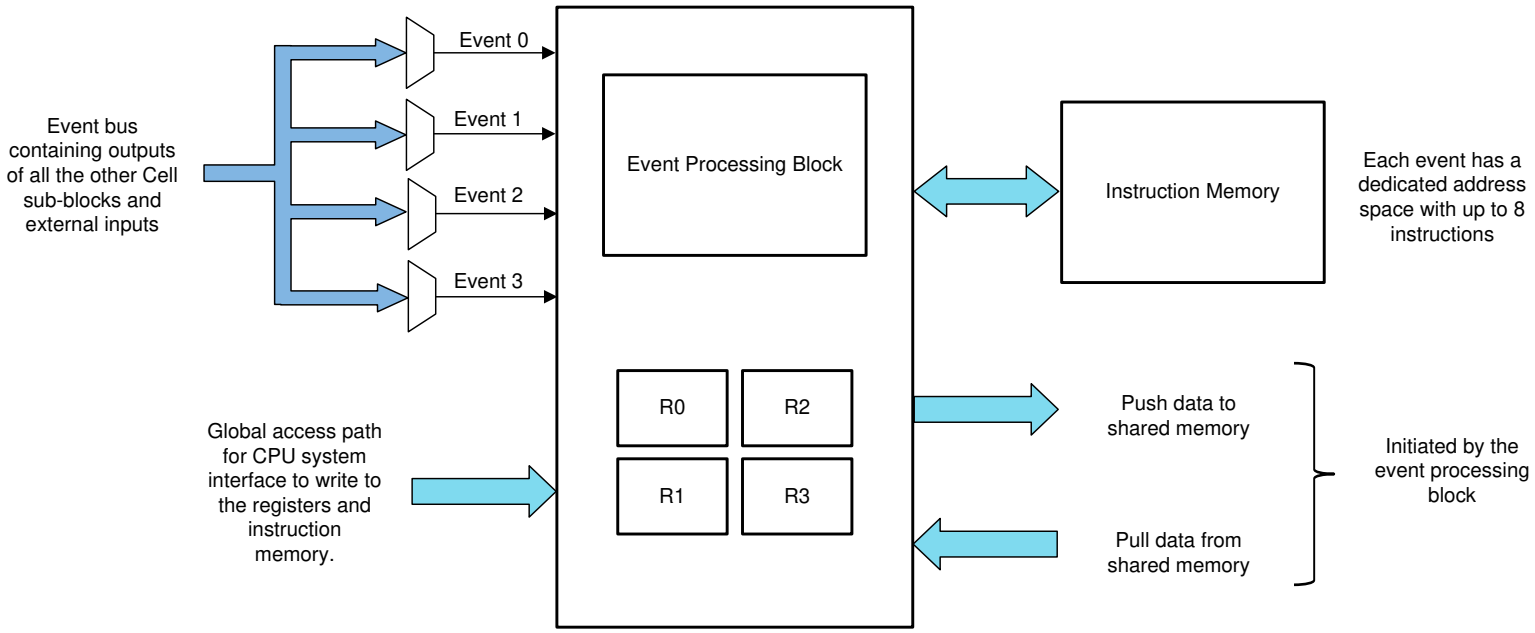



**Figure 26-15. High Level Controller Block**



Copyright © 2024 Texas Instruments Incorporated

#### 26.4.6.1 High Level Controller Events

Each of the four HLC events has a dedicated address from which instructions are executed. Events are selected from the set of signals listed in Table 26-8. The lowest numbered event (Event 0) has the highest priority, and the highest numbered event (Event 3) has the lowest priority.

**Table 26-8. HLC Event List**

| Index | HLC Event Mux |
|---|---|
| 0 | Always 0 |
| 1 | COUNTER_0 MATCH2 |
| 2 | COUNTER_0 ZERO |
| 3 | COUNTER_0 MATCH1 |
| 4 | FSM_0 STATE_BIT_0 |
| 5 | FSM_0 STATE_BIT_1 |
| 6 | FSM_0 LUT output |
| 7 | LUT4_0 output |
| 8 | Always 1 |
| 9 | COUNTER_1 MATCH2 |
| 10 | COUNTER_1 ZERO |
| 11 | COUNTER_1 MATCH1 |
| 12 | FSM_1 STATE_BIT_0 |
| 13 | FSM_1 STATE_BIT_1 |
| 14 | FSM_1 LUT output |
| 15 | LUT4_1 output |
| 16 | Always ‘0’ |
| 17 | COUNTER_2 MATCH2 |
| 18 | COUNTER_2 ZERO |
| 19 | COUNTER_2 MATCH1 |
| 20 | FSM_2 STATE_BIT_0 |
| 21 | FSM_2 STATE_BIT_1 |
| 22 | FSM_2 LUT output |
| 23 | LUT4_2 output |
| 24 | External Input 0 |
| 25 | External Input 1 |
| 26 | External Input 2 |
| 27 | External Input 3 |
| 28 | External Input 4 |
| 29 | External Input 5 |
| 30 | External Input 6 |
| 31 | External Input 7 |



Copyright © 2024 Texas Instruments Incorporated

#### 26.4.6.2 High Level Controller Instructions

The instruction memory supports up to eight instructions per event. Each instruction sequence gets triggered on the rising edge of the corresponding event.

The HLC memory supports up to eight instructions per event, starting at the beginning of the fixed address range shown in Table 26-9. An instruction sequence is triggered on the rising edge of the corresponding event. If two or more events occur simultaneously, the associated instruction sequences each are executed sequentially in priority order.

**Table 26-9. HLC Instruction Address Ranges**

| Address | Instructions for |
|---|---|
| 00000 to 00111 | Event 0 |
| 01000 to 01111 | Event 1 |
| 10000 to 10111 | Event 2 |
| 11000 to 11111 | Event 3 |

The HLC instruction format is shown in Table 26-10.

**Table 26-10. HLC Instruction Format**

| Last Instruction Bit | 5-Bit Opcode | 3-Bit Source | 3-Bit Destination |
|---|---|---|---|
| This bit when set to 1 stops execution after the current instruction. | MOV 00000<br>MOV_T1 00001<br>MOV_T2 00010<br>PUSH 00011<br>PULL 00100<br>ADD 00101<br>SUB 00110<br>INTR 00111 | Source can be R0, R1, R2, R3, C0, C1, C2. | Destination can be R0, R1, R2, R3, C0, C1, C2.<br>Note that for ADD/SUB instructions, only R0, R1, R2, or R3 can be the destination. |

R0, R1, R2, and R3 are four 32-bit general-purpose registers in the HLC. C0, C1, and C2 are three counter registers present in the CLB tile. <Src> is used to indicate the source and <Dest> is used to indicate the destination. Table 26-11 describes the HLC instructions.

**Table 26-11. HLC Instruction Description**

| Instruction | Description |
|---|---|
| ADD <Src>, <Dest> | This instruction performs an unsigned 32-bit addition. <Dest> = <Dest> + <Src>. The <Src> can be R0, R1, R2, R3, C0, C1, or C2. The <Dest> can only be R0, R1, R2, or R3. |
| INTR <6-bit constant> | This instruction flags an interrupt through the CPU interface. The 6-bit constant is stored in the interrupt flag register CLB_INTR_TAG_REG. If multiple INTR instructions are called consecutively, only the first one has an effect. When multiple INTR calls are needed, each can be separated by other HLC instructions to make sure the interrupt calls take effect. |
| MOV <Src>, <Dest> | This instruction moves <Src> to <Dest>. Both <Src> and <Dest> can be any of R0, R1, R2, R3, C0, C1, or C2. The COUNT_EVENT_CTRL_x bit must be configured to load (that is, 0) for indirect loads and HLC loads of the counter to take effect. |
| MOV_T1 <Src>, <Dest> | This instruction moves <Src> to the Match1 register of the <Dest> counter. <Src> can be any of the registers R0, R1, R2, R3, or the counter values associated with C0, C1, or C2. <Dest> is the Match1 register of any of the counters C0, C1, or C2. Examples are:<br>• This instruction moves the count value in C1 into register R0:<br>MOV_T1 C1 R0<br>• This instruction moves the value in R2 into the Match1 register of counter C0:<br>MOV_T1 R2 C0 |
| MOV_T2 <Src>, <Dest> | This instruction is similar to MOV_T1. The instruction moves <Src> to the Match2 register of the <Dest> counter. <Src> can be any of the registers R0, R1, R2, R3, or the counter values associated with C0, C1, or C2. <Dest> is the Match2 register of any of the counters C0, C1, or C2. |



Copyright © 2024 Texas Instruments Incorporated

**Table 26-11. HLC Instruction Description (continued)**

| Instruction | Description |
|---|---|
| PULL <Dest> | This instruction transfers data from the data exchange pull memory buffer in the CPU interface to the <Dest> register. <Dest> can be any of R0, R1, R2, or R3. The PULL instruction is used as seen from the High Level Controller and a PULL operation reads (pulls) data from an internal 4-word FIFO.<br>**Note**<br>Back-to-back PUSH/PULL instructions cannot be used, if using more than one HLC EVENT channel. |
| PUSH <Src> | This instruction transfers data from <Src> to the data exchange push memory buffer in the CPU interface. <Src> can be any of R0, R1, R2, R3, C0, C1, or C2. The PUSH instruction is used as seen from the High Level Controller and pushes data into an internal 4 word FIFO. |
| SUB <Src>, <Dest> | This instruction performs an unsigned 32-bit subtraction. <Dest> = <Dest> - <Src>. The <Src> can be R0, R1, R2, R3, C0, C1, or C2. The <Dest> can only be R0, R1, R2, or R3. |

MOV, MOV_T1, MOV_T2, ADD, SUB, and INTR instructions take one cycle to execute. PUSH and PULL require two cycles to execute. Note that the PUSH and PULL instructions are pipeline protected, meaning that a register can be used immediately after a PUSH/PULL to that register.

For multiple events triggered simultaneously, if the last instruction in the higher priority event is a PUSH or a PULL, there is an additional cycle delay between the end of the higher priority event and the start of the next event. If the last instruction is not a PUSH or PULL, then there is no cycle delay between the events.

If there is the possibility of more than one event coming into the HLC, then PUSH/PULL instructions must not be placed back-to-back. It is necessary to re-order the HLC instructions such that every PUSH and PULL is followed by a non-PUSH/PULL instruction. If only one event can enter the HLC at any time, then back-to-back PUSH/PULL is not an issue.

#### 26.4.6.3 <Src> and <Dest>

Three bits are used to encode the <Src> and <Dest> registers as shown in Table 26-12.

**Table 26-12. HLC Register Encoding**

| Bits | Register |
|---|---|
| 000 | R0 |
| 001 | R1 |
| 010 | R2 |
| 011 | R3 |
| 100 | C0 |
| 101 | C1 |
| 110 | C2 |



Copyright © 2024 Texas Instruments Incorporated

#### 26.4.6.4 Operation of the PUSH and PULL Instructions (Overflow and Underflow Detection)

The PUSH and PULL operations of the HLC are intended for data exchange with the host system. There are separate FIFO buffers for PUSH and PULL operations. For example, a series of PUSH operations write to successive locations in a linearly mapped-memory buffer. The PUSH buffer and the PULL buffer are mapped at address offsets shown in the *Registers* section.

The CPU can read from and write to the PUSH and PULL buffers, respectively, to exchange data with the HLC. Data pushed by the HLC is read by the CPU from the PUSH buffers. Data sent from the CPU to the HLC is written by the CPU to the PULL buffer and is read by the HLC using the PULL instruction.

Refer to *clb_ex13_push_pull* for guidance on properly using the PUSH and PULL buffers. To make use of one of the CLB inputs as a GPREG, have this input indicate when data is written to the FIFO by the CPU.

There are separate PUSH and PULL address pointers that increment each time the HLC performs a PUSH or PULL operation. These address pointers are also memory-mapped so that the CPU can determine the value. These address pointers are also writable and can be reset by the CPU at any time.

Overflow and underflow detection is done by simply reading the values of the PUSH and PULL address pointers.

In the CLB module of the device, the depth of the PUSH and PULL FIFOs is four 32-bit words each. If the CPU starts a fresh data transfer to the PULL buffers and sees the address pointer greater than four, then an underflow has occurred since the HLC has pulled more data than the number of words written by the CPU into the buffer.

**Note**

Back-to-back PUSH and PULL instructions (in either order) do not work, if you have more than one HLC event enabled.

## 26.5 CPU Interface

### *26.5.1 Register Description*

There are three classes of registers that are used to control and configure the CLB tile. This specification only describes the offset addresses of the registers. The absolute register addresses are different for each CLB tile. The three instances of the various blocks (LUT4, FSM, and Counter Block) are numbered 0, 1, and 2.

- **Logic configuration registers (0x000 – 0x0FF):** These registers control the core reconfiguration logic for the tile. All registers in this group are EALLOW protected and also protected by the LOCK register.
- **Top level control registers (0x100 – 0x1FF):** These registers are used for top level and device related control of the CLB. These registers typically control mux selects for inputs, global enables, and so forth, and are accessible by normal memory mapped access. Some of these registers have EALLOW and LOCK protection.
- **Data exchange registers (0x200 – 0x3FF):** These registers are used to exchange data between the CLB and the rest of the device. The registers are accessible by normal memory-mapped access and no EALLOW or LOCK protection exists.

**Note**

EALLOW protection means that the write access to the register is enabled only when the EALLOW instruction has been executed prior to the write access. The complementary EDIS instruction disables access to all registers protected in this way. For more information, see Section 26.8.



Copyright © 2024 Texas Instruments Incorporated

### 26.5.2 Non-Memory Mapped Registers

The memory-mapped CLB registers are described later in this chapter; however, many of the CLB resources including counters, the instruction memory of the High Level Controller, and the HLC general-purpose registers (R0 through R3) are only indirectly accessible through a local interface bus and are not memory-mapped. These registers are accessible through the two memory-mapped registers CLB_LOAD_DATA and CLB_LOAD_ADDR. Note that the general-purpose registers R0 through R3 must only be written to during configuration-time and are not intended to be written to during run-time. Writes during run-time can lead to unexpected behavior. If run-time data exchange is desired, refer to Section 26.4.6.4.

Load the data to be written into the CLB_LOAD_DATA register, then load the appropriate address into CLB_LOAD_ADDR to determine where this data is written. Writing a 1 to bit position 0 in the CLB_LOAD_EN register then causes an internal write operation to be triggered. The address allocation for the CLB_LOAD_ADDR register is shown in Table 26-13.

**Note**

The COUNT_EVENT_CTRL_x bit must be configured to load (that is, 0) for indirect loads and for HLC loads of the counter to take effect.

**Table 26-13. Non-Memory Mapped Register Addresses**

| Address (Binary) | Resource |
|---|---|
| 000000 to 000010 | Counter 0 to 2 Load value |
| 000100 to 000110 | Counter 0 to 2 Match1 value |
| 001000 to 001010 | Counter 0 to 2 Match2 value |
| 001100 to 001111 | R0 to R3 of High Level Controller |
| 100000 to 100111 | Instructions for Event 0 |
| 101000 to 101111 | Instructions for Event 1 |
| 110000 to 110111 | Instructions for Event 2 |
| 111000 to 111111 | Instructions for Event 3 |

Use the following steps to load the value 0x11223344 into the general purpose R0 register:

1. Write 0x11223344 to CLB_LOAD_DATA.
2. Write 0xc to CLB_LOAD_ADDR.
3. Write 0x1 to CLB_LOAD_EN.

**Note**

Even though HLC registers are accessible by the CPU, your application code needs to make sure that no other CLB internal logics are updating the same HLC register at the same time, causing a race condition.

## 26.6 DMA Access

The DMA does not have access to the CLB memory-mapped registers, including the PUSH and PULL FIFO registers. For more information, refer to Section 26.8.



Copyright © 2024 Texas Instruments Incorporated

## 26.7 Software

### *26.7.1 CLB Examples*

NOTE: These examples are located in the C2000Ware installation at the following location:
C2000Ware_VERSION#/driverlib/DEVICE_GPN/examples/CORE_IF_MULTICORE/clb

Cloud access to these examples is available at the following link: dev.ti.com C2000Ware Examples.

#### 26.7.1.1 CLB Empty Project

FILE: empty.c

For the detailed description of this example, please refer to:
C2000Ware_PATH\utilities\clb_tool\clb_syscfg\doc\CLB Tool Users Guide.pdf

#### 26.7.1.2 CLB Combinational Logic

FILE: clb_ex1_combinatorial_logic.c

For the detailed description of this example, please refer to:
C2000Ware_PATH\utilities\clb_tool\clb_syscfg\doc\CLB Tool Users Guide.pdf

The objective of this example is to prevent simultaneous high or low outputs on a PWM pair. PWM modules 1 and 2 are configured to generate identical waveforms based on a fixed frequency up-count mode.

#### 26.7.1.3 CLB GPIO Input Filter

FILE: clb_ex2_gpio_input_filter.c

For the detailed description of this example, please refer to:
C2000Ware_PATH\utilities\clb_tool\clb_syscfg\doc\CLB Tool Users Guide.pdf

This example demonstrates use of finite state machines (FSMs) and counters to implement a simple ‘glitch’ filter which might, for example, be applied to an incoming GPIO signal to remove unwanted short duration pulses.

#### 26.7.1.4 CLB Auxilary PWM

FILE: clb_ex3_auxiliary_pwm.c

For the detailed description of this example, please refer to:
C2000Ware_PATH\utilities\clb_tool\clb_syscfg\doc\CLB Tool Users Guide.pdf

This example configures a CLB tile as an auxiliary PWM generator. The example uses combinatorial logic (LUTs), state machines (FSMs), counters, and the high level controller (HLC) to demonstrate the PWM output generation capabilities using CLB.

#### 26.7.1.5 CLB PWM Protection

FILE: clb_ex4_pwm_protection.c

For the detailed description of this example, please refer to:
C2000Ware_PATH\utilities\clb_tool\clb_syscfg\doc\CLB Tool Users Guide.pdf

This example extends the features of example 1 to ensure an active high complementary pair PWM configuration always operates with a minimum value of dead-band irrespective of how the generating PWM module is configured. The example illustrates the configuration of four separate PWM tiles to implement PWM protection on four PWM modules. The outputs of PWM modules 1 to 4 are operated on by CLB tiles 1 to 4, respectively.

#### 26.7.1.6 CLB Event Window

FILE: clb_ex5_event_window.c

For the detailed description of this example, please refer to:
C2000Ware_PATH\utilities\clb_tool\clb_syscfg\doc\CLB Tool Users Guide.pdf



Copyright © 2024 Texas Instruments Incorporated

This example uses the counter, FSM, and HLC sub-modules of the CLB to implement an event timing feature which detects whether an interrupt service routine takes too long to respond to an interrupt. The example configures four PWM modules to operate in up-count mode and generate a low-to-high edge on a timer zero match event. The zero match event also triggers a PWM ISR which, for the purposes of this example, contains a dummy payload of variable length. At the end of the ISR, a write operation takes place to a CLB GP register to indicate the ISR has ended.

#### 26.7.1.7 CLB Signal Generator

FILE: clb_ex6_siggen.c

For the detailed description of this example, please refer to:
C2000Ware_PATH\utilities\clb_tool\clb_syscfg\doc\CLB Tool Users Guide.pdf

This example uses CLB1 to generate a rectangular wave and CLB2 to check the rectangular wave generated by CLB1 doesn't exceed the defined duty cycle and period limits.

#### 26.7.1.8 CLB State Machine

FILE: clb_ex7_state_machine.c

For the detailed description of this example, please refer to:
C2000Ware_PATH\utilities\clb_tool\clb_syscfg\doc\Designing With the C2000 CLB.pdf This application report describes the process of creating this CLB example and can be used as guidance on desiging custom logic with the CLB. This example uses all submodules inside a CLB TILE in order to implement a complete system.

#### 26.7.1.9 CLB External Signal AND Gate

FILE: clb_ex8_external_signal_AND_gate.c

For the detailed description of this example, please refer to:
C2000Ware_PATH\utilities\clb_tool\clb_syscfg\doc\CLB Tool Users Guide.pdf

In this example, two external signals from two GPIOs are passed through the Input X-BAR and the CLB X-BAR to the CLB TILE. Inside the CLB module these two signals are ANDED. The output of the AND gate is then exported to a GPIO, using Output X-BAR.

#### 26.7.1.10 CLB Timer

FILE: clb_ex9_timer.c

For the detailed description of this example, please refer to:
C2000Ware_PATH\utilities\clb_tool\clb_syscfg\doc\CLB Tool Users Guide.pdf

In this example, a COUNTER module is used to create timed events. The use of the GP Register is shown. Through setting/clearing the bits in the GP register, the timer is started, stopped or changes direction. The output of the timer event (1-clock cycle) is exported to a GPIO. Interrupts are generated from the timer event using the HLC module. A GPIO is also toggled inside the CLB ISR. The indirect CLB register access is used to update the timer's event match value and the active counter register to modify the frequency of the timer.

#### 26.7.1.11 CLB Timer Two States

FILE: clb_ex10_timer_two_states.c

For the detailed description of this example, please refer to:
C2000Ware_PATH\utilities\clb_tool\clb_syscfg\doc\CLB Tool Users Guide.pdf

In this example, the timer is setup the same as the previous example. The difference is the use of the FSM submodule to toggle the output of the CLB which is then exported to a GPIO. The FSM module acts as a single bit memory block. Interrupts are setup in the same format as the previous example. The interrupt delay of the CLB can be seen by comparing the output of the CLB and the GPIO toggled in the ISR.

#### 26.7.1.12 CLB Interrupt Tag

FILE: clb_ex11_interrupt_tag.c



Copyright © 2024 Texas Instruments Incorporated

For the detailed description of this example, please refer to:
C2000Ware_PATH\utilities\clb_tool\clb_syscfg\doc\CLB Tool Users Guide.pdf

In this example, a timer is setup with two different match values. These two events are used by the HLC submodule to generate interrupts. The interrupt TAG is used to differentiate between the interrupt generated due to the match1 event of the CLB counter and the match2 event of the CLB counter.

#### 26.7.1.13 CLB Output Intersect

FILE: clb_ex12_output_intersect.c

For the detailed description of this example, please refer to:
C2000Ware_PATH\utilities\clb_tool\clb_syscfg\doc\CLB Tool Users Guide.pdf

In this example, the CLB module is set up the same as the external_AND_gate example. However, instead of the output being exported to the GPIO using Output X-BAR, the output is exported to the GPIO by replacing the output of ePWM1. This is done by configuring the GPIO for EPWM1A output, followed by enabling output intersection.

#### 26.7.1.14 CLB PUSH PULL

FILE: clb_ex13_push_pull.c

For the detailed description of this example, please refer to:
C2000Ware_PATH\utilities\clb_tool\clb_syscfg\doc\CLB Tool Users Guide.pdf

In this example, the use of the PUSH-PULL interface is shown. Multiple COUNTER submodules, HLC submodule, FSM submodules, and OUTLUT submodules are used. The PUSH-PULL interface is used alongside the GP register to update the COUNTER submodules' event frequencies.

#### 26.7.1.15 CLB Multi Tile

FILE: clb_ex14_multi_tile.c

For the detailed description of this example, please refer to:
C2000Ware_PATH\utilities\clb_tool\clb_syscfg\doc\CLB Tool Users Guide.pdf

In this example the output of a CLB TILE is passed to the input of another CLB TILE. The output of the second CLB TILE is then exported to a GPIO, showcasing how two CLB TILEs can be used in series.

#### 26.7.1.16 CLB Tile to Tile Delay

FILE: clb_ex15_tile_to_tile_delay.c

For the detailed description of this example, please refer to:
C2000Ware_PATH\utilities\clb_tool\clb_syscfg\doc\CLB Tool Users Guide.pdf

In this example the output of a GPIO is taken into the CLB TILE through INPUT XBAR and the CLB XBAR. The signal is forwarded by the TILE to the next TILE. This time the signal only goes through the CLB XBAR and NOT the Input XBAR. This is done to show that delays are added when the signals are passed from TILE to TILE and the delay is NOT characterized. The user should always avoid passing signals with timing requirements between tiles. The COUNTER modules inside the CLBs will count the amount of delay in cycles.

#### 26.7.1.17 CLB based One-shot PWM

FILE: clb_ex17_one_shot_pwm.c

For the detailed description of this example, please refer to :
C2000Ware_PATH\utilities\clb_tool\clb_syscfg\doc\CLB Tool Users Guide.pdf

#### 26.7.1.18 CLB Trip Zone Timestamp

FILE: clb_ex29_timestamp.c

This example displays how to timestamp interrupts generated by the CLB. An interrupt is generated when ePWM1 is tripped.



Copyright © 2024 Texas Instruments Incorporated

ePWM1 is configured to be interrupted by TZ1 and TZ2, both one shot trip sources.

The CLB is configured as follows:

- COUNTER0 and COUNTER1 continually count when the program begins.
- COUNTER0 timestamps TZ1 and COUNTER1 timestamps TZ2.
- COUNTER2 increments once when COUNTER0/COUNTER1 overflows using LUT2.
- FSM0/1 are configured to sync counters and stop COUNTER0/1 when an interrupt is received.
- TZ1 (GPIO12) and TZ2 (GPIO13) are routed as inputs through CLBXBAR.
- BOUNDARY.boundaryInput0 denotes TZ1. On rising edge, HLC issues an interrupt with tag 12.
- BOUDNARY.in1 denotes TZ2. On rising edge, HLC issues an interrupt with tag 13.
- BOUNDARY.boundaryInput7 serves as a simultaneous enable for COUNTER0/1 to begin counting.

TZ1 is tripped when GPIO12 is connected to GND. TZ2 is tripped when GPIO13 is connected to GND. When an interrupt occurs, the interrupt handler determines the initial trip source and stores this value in a variable 'initialTripZone'.

View these variables in Debug Expressions tab:

initialTripZone: stores the first TZ to have been tripped tz1Counter64bit: stores the counter value at the instant that TZ1 is tripped. tz2Counter64bit: stores the counter value at the instant that TZ2 is tripped.

## 26.8 CLB Registers

This section describes the Configurable Logic Block Registers.

### *26.8.1 CLB Base Addresses*

**Table 26-14. CLB Base Address Table (C28)**

| Bit Field Name | | DriverLib Name | Base Address |
|---|---|---|---|
| Instance | Structure | | |
| Clb1LogicCfgRegs | CLB_LOGIC_CONFIG_REGS | CLB1_LOGICCFG_BASE | 0x0000_3000 |
| Clb1LogicCtrlRegs | CLB_LOGIC_CONTROL_REGS | CLB1_LOGICCTRL_BASE | 0x0000_3100 |
| Clb1DataExchRegs | CLB_DATA_EXCHANGE_REGS | CLB1_DATAEXCH_BASE | 0x0000_3200 |
| Clb2LogicCfgRegs | CLB_LOGIC_CONFIG_REGS | CLB2_LOGICCFG_BASE | 0x0000_3400 |
| Clb2LogicCtrlRegs | CLB_LOGIC_CONTROL_REGS | CLB2_LOGICCTRL_BASE | 0x0000_3500 |
| Clb2DataExchRegs | CLB_DATA_EXCHANGE_REGS | CLB2_DATAEXCH_BASE | 0x0000_3600 |
| Clb3LogicCfgRegs | CLB_LOGIC_CONFIG_REGS | CLB3_LOGICCFG_BASE | 0x0000_3800 |
| Clb3LogicCtrlRegs | CLB_LOGIC_CONTROL_REGS | CLB3_LOGICCTRL_BASE | 0x0000_3900 |
| Clb3DataExchRegs | CLB_DATA_EXCHANGE_REGS | CLB3_DATAEXCH_BASE | 0x0000_3A00 |
| Clb4LogicCfgRegs | CLB_LOGIC_CONFIG_REGS | CLB4_LOGICCFG_BASE | 0x0000_3C00 |
| Clb4LogicCtrlRegs | CLB_LOGIC_CONTROL_REGS | CLB4_LOGICCTRL_BASE | 0x0000_3D00 |
| Clb4DataExchRegs | CLB_DATA_EXCHANGE_REGS | CLB4_DATAEXCH_BASE | 0x0000_3E00 |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.2 CLB_LOGIC_CONFIG_REGS Registers

Table 26-15 lists the memory-mapped registers for the CLB_LOGIC_CONFIG_REGS registers. All register offset addresses not listed in Table 26-15 should be considered as reserved locations and the register contents should not be modified.

**Table 26-15. CLB_LOGIC_CONFIG_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 2h | CLB_COUNT_RESET | Counter Block RESET | EALLOW, LOCK | Go |
| 4h | CLB_COUNT_MODE_1 | Counter Block MODE_1 | EALLOW, LOCK | Go |
| 6h | CLB_COUNT_MODE_0 | Counter Block MODE_0 | EALLOW, LOCK | Go |
| 8h | CLB_COUNT_EVENT | Counter Block EVENT | EALLOW, LOCK | Go |
| Ah | CLB_FSM_EXTRA_IN0 | FSM Extra EXT_IN0 | EALLOW, LOCK | Go |
| Ch | CLB_FSM_EXTERNAL_IN0 | FSM EXT_IN0 | EALLOW, LOCK | Go |
| Eh | CLB_FSM_EXTERNAL_IN1 | FSM_EXT_IN1 | EALLOW, LOCK | Go |
| 10h | CLB_FSM_EXTRA_IN1 | FSM Extra_EXT_IN1 | EALLOW, LOCK | Go |
| 12h | CLB_LUT4_IN0 | LUT4_0/1/2 IN0 input source | EALLOW, LOCK | Go |
| 14h | CLB_LUT4_IN1 | LUT4_0/1/2 IN1 input source | EALLOW, LOCK | Go |
| 16h | CLB_LUT4_IN2 | LUT4_0/1/2 IN2 input source | EALLOW, LOCK | Go |
| 18h | CLB_LUT4_IN3 | LUT4_0/1/2 IN3 input source | EALLOW, LOCK | Go |
| 1Ch | CLB_FSM_LUT_FN1_0 | LUT function for FSM Unit 1 and Unit 0 | EALLOW, LOCK | Go |
| 1Eh | CLB_FSM_LUT_FN2 | LUT function for FSM Unit 2 | EALLOW, LOCK | Go |
| 20h | CLB_LUT4_FN1_0 | LUT function for LUT4 block of Unit 1 and 0 | EALLOW, LOCK | Go |
| 22h | CLB_LUT4_FN2 | LUT function for LUT4 block of Unit 2 | EALLOW, LOCK | Go |
| 24h | CLB_FSM_NEXT_STATE_0 | FSM Next state equations for Unit 0 | EALLOW, LOCK | Go |
| 26h | CLB_FSM_NEXT_STATE_1 | FSM Next state equations for Unit 1 | EALLOW, LOCK | Go |
| 28h | CLB_FSM_NEXT_STATE_2 | FSM Next state equations for Unit 2 | EALLOW, LOCK | Go |
| 2Ah | CLB_MISC_CONTROL | Static controls for Ctr,FSM | EALLOW, LOCK | Go |
| 2Ch | CLB_OUTPUT_LUT_0 | Inp Sel, LUT fns for Out0 | EALLOW, LOCK | Go |
| 2Eh | CLB_OUTPUT_LUT_1 | Inp Sel, LUT fns for Out1 | EALLOW, LOCK | Go |
| 30h | CLB_OUTPUT_LUT_2 | Inp Sel, LUT fns for Out2 | EALLOW, LOCK | Go |
| 32h | CLB_OUTPUT_LUT_3 | Inp Sel, LUT fns for Out3 | EALLOW, LOCK | Go |
| 34h | CLB_OUTPUT_LUT_4 | Inp Sel, LUT fns for Out4 | EALLOW, LOCK | Go |
| 36h | CLB_OUTPUT_LUT_5 | Inp Sel, LUT fns for Out5 | EALLOW, LOCK | Go |
| 38h | CLB_OUTPUT_LUT_6 | Inp Sel, LUT fns for Out6 | EALLOW, LOCK | Go |
| 3Ah | CLB_OUTPUT_LUT_7 | Inp Sel, LUT fns for Out7 | EALLOW, LOCK | Go |
| 3Ch | CLB_HLC_EVENT_SEL | Event Selector register for the High Level controller | EALLOW, LOCK | Go |

Complex bit access types are encoded to fit into small table cells. Table 26-16 shows the codes that are used for access types in this section.

**Table 26-16. CLB_LOGIC_CONFIG_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| R-0 | R<br>-0 | Read<br>Returns 0s |
| Write Type | | |
| W | W | Write |



Copyright © 2024 Texas Instruments Incorporated

**Table 26-16. CLB_LOGIC_CONFIG_REGS Access Type Codes (continued)**

| Access Type | Code | Description |
|---|---|---|
| W1C | W<br>1C | Write<br>1 to clear |
| 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. |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.2.1 CLB_COUNT_RESET Register (Offset = 2h) [Reset = 00000000h]

CLB_COUNT_RESET is shown in Figure 26-16 and described in Table 26-17.

Return to the Summary Table.

Counter Block RESET

**Figure 26-16. CLB_COUNT_RESET Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | | | SEL_2 | | | | | SEL_1 | | | | | SEL_0 | | | | |
| R/W1C-0h | | | | | | | | | | | | | | | | | R/W-0h | | | | | R/W-0h | | | | | R/W-0h | | | | |

**Table 26-17. CLB_COUNT_RESET Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-15 | RESERVED | R/W1C | 0h | Reserved |
| 14-10 | SEL_2 | R/W | 0h | Counter reset select inputs for unit 2. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |
| 9-5 | SEL_1 | R/W | 0h | Counter reset select inputs for unit 1. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |
| 4-0 | SEL_0 | R/W | 0h | Counter reset select inputs for unit 0. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.2.2 CLB_COUNT_MODE_1 Register (Offset = 4h) [Reset = 00000000h]

CLB_COUNT_MODE_1 is shown in Figure 26-17 and described in Table 26-18.

Return to the Summary Table.

Counter Block MODE_1

**Figure 26-17. CLB_COUNT_MODE_1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | | | SEL_2 | | | | | SEL_1 | | | | | SEL_0 | | | | |
| R/W1C-0h | | | | | | | | | | | | | | | | | R/W-0h | | | | | R/W-0h | | | | | R/W-0h | | | | |

**Table 26-18. CLB_COUNT_MODE_1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-15 | RESERVED | R/W1C | 0h | Reserved |
| 14-10 | SEL_2 | R/W | 0h | Counter MODE_1 select inputs for unit 2. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |
| 9-5 | SEL_1 | R/W | 0h | Counter MODE_1 select inputs for unit 1. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |
| 4-0 | SEL_0 | R/W | 0h | Counter MODE_1 select inputs for unit 0. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.2.3 CLB_COUNT_MODE_0 Register (Offset = 6h) [Reset = 00000000h]

CLB_COUNT_MODE_0 is shown in Figure 26-18 and described in Table 26-19.

Return to the Summary Table.

Counter Block MODE_0

**Figure 26-18. CLB_COUNT_MODE_0 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | | | SEL_2 | | | | | SEL_1 | | | | | SEL_0 | | | | |
| R/W1C-0h | | | | | | | | | | | | | | | | | R/W-0h | | | | | R/W-0h | | | | | R/W-0h | | | | |

**Table 26-19. CLB_COUNT_MODE_0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-15 | RESERVED | R/W1C | 0h | Reserved |
| 14-10 | SEL_2 | R/W | 0h | Counter MODE_0 select inputs for unit 2. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |
| 9-5 | SEL_1 | R/W | 0h | Counter MODE_0 select inputs for unit 1. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |
| 4-0 | SEL_0 | R/W | 0h | Counter MODE_0 select inputs for unit 0. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.2.4 CLB_COUNT_EVENT Register (Offset = 8h) [Reset = 00000000h]

CLB_COUNT_EVENT is shown in Figure 26-19 and described in Table 26-20.

Return to the Summary Table.

Counter Block EVENT

**Figure 26-19. CLB_COUNT_EVENT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | | | SEL_2 | | | | | SEL_1 | | | | | SEL_0 | | | | |
| R/W1C-0h | | | | | | | | | | | | | | | | | R/W-0h | | | | | R/W-0h | | | | | R/W-0h | | | | |

**Table 26-20. CLB_COUNT_EVENT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-15 | RESERVED | R/W1C | 0h | Reserved |
| 14-10 | SEL_2 | R/W | 0h | Counter event select inputs for unit 2. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |
| 9-5 | SEL_1 | R/W | 0h | Counter event select inputs for unit 1. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |
| 4-0 | SEL_0 | R/W | 0h | Counter event select inputs for unit 0. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.2.5 CLB_FSM_EXTRA_IN0 Register (Offset = Ah) [Reset = 00000000h]

CLB_FSM_EXTRA_IN0 is shown in Figure 26-20 and described in Table 26-21.

Return to the Summary Table.

FSM Extra EXT_IN0

**Figure 26-20. CLB_FSM_EXTRA_IN0 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | | | SEL_2 | | | | | SEL_1 | | | | | SEL_0 | | | | |
| R/W1C-0h | | | | | | | | | | | | | | | | | R/W-0h | | | | | R/W-0h | | | | | R/W-0h | | | | |

**Table 26-21. CLB_FSM_EXTRA_IN0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-15 | RESERVED | R/W1C | 0h | Reserved |
| 14-10 | SEL_2 | R/W | 0h | FSM block extra external IN0 select inputs for unit 2. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |
| 9-5 | SEL_1 | R/W | 0h | FSM block extra external IN0 select inputs for unit 1. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |
| 4-0 | SEL_0 | R/W | 0h | FSM block extra external IN0 select inputs for unit 0. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.2.6 CLB_FSM_EXTERNAL_IN0 Register (Offset = Ch) [Reset = 00000000h]

CLB_FSM_EXTERNAL_IN0 is shown in Figure 26-21 and described in Table 26-22.

Return to the Summary Table.

FSM EXT_IN0

**Figure 26-21. CLB_FSM_EXTERNAL_IN0 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | | | SEL_2 | | | | | SEL_1 | | | | | SEL_0 | | | | |
| R/W1C-0h | | | | | | | | | | | | | | | | | R/W-0h | | | | | R/W-0h | | | | | R/W-0h | | | | |

**Table 26-22. CLB_FSM_EXTERNAL_IN0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-15 | RESERVED | R/W1C | 0h | Reserved |
| 14-10 | SEL_2 | R/W | 0h | FSM block EXT_IN0 select input for unit 2. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |
| 9-5 | SEL_1 | R/W | 0h | FSM block EXT_IN0 select input for unit 1. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |
| 4-0 | SEL_0 | R/W | 0h | FSM block EXT_IN0 select input for unit 0. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.2.7 CLB_FSM_EXTERNAL_IN1 Register (Offset = Eh) [Reset = 00000000h]

CLB_FSM_EXTERNAL_IN1 is shown in Figure 26-22 and described in Table 26-23.

Return to the Summary Table.

FSM_EXT_IN1

**Figure 26-22. CLB_FSM_EXTERNAL_IN1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | | | SEL_2 | | | | | SEL_1 | | | | | SEL_0 | | | | |
| R/W1C-0h | | | | | | | | | | | | | | | | | R/W-0h | | | | | R/W-0h | | | | | R/W-0h | | | | |

**Table 26-23. CLB_FSM_EXTERNAL_IN1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-15 | RESERVED | R/W1C | 0h | Reserved |
| 14-10 | SEL_2 | R/W | 0h | FSM block EXT_IN1 select input for unit 2. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |
| 9-5 | SEL_1 | R/W | 0h | FSM block EXT_IN1 select input for unit 1. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |
| 4-0 | SEL_0 | R/W | 0h | FSM block EXT_IN1 select input for unit 0. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.2.8 CLB_FSM_EXTRA_IN1 Register (Offset = 10h) [Reset = 00000000h]

CLB_FSM_EXTRA_IN1 is shown in Figure 26-23 and described in Table 26-24.

Return to the Summary Table.

FSM Extra_EXT_IN1

**Figure 26-23. CLB_FSM_EXTRA_IN1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | | | SEL_2 | | | | | SEL_1 | | | | | SEL_0 | | | | |
| R/W1C-0h | | | | | | | | | | | | | | | | | R/W-0h | | | | | R/W-0h | | | | | R/W-0h | | | | |

**Table 26-24. CLB_FSM_EXTRA_IN1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-15 | RESERVED | R/W1C | 0h | Reserved |
| 14-10 | SEL_2 | R/W | 0h | FSM block extra external IN1 select inputs for unit 2. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |
| 9-5 | SEL_1 | R/W | 0h | FSM block extra external IN1 select inputs for unit 1. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |
| 4-0 | SEL_0 | R/W | 0h | FSM block extra external IN1 select inputs for unit 0. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.2.9 CLB_LUT4_IN0 Register (Offset = 12h) [Reset = 00000000h]

CLB_LUT4_IN0 is shown in Figure 26-24 and described in Table 26-25.

Return to the Summary Table.

LUT4_0/1/2 IN0 input source

**Figure 26-24. CLB_LUT4_IN0 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | | | SEL_2 | | | | | SEL_1 | | | | | SEL_0 | | | | |
| R/W1C-0h | | | | | | | | | | | | | | | | | R/W-0h | | | | | R/W-0h | | | | | R/W-0h | | | | |

**Table 26-25. CLB_LUT4_IN0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-15 | RESERVED | R/W1C | 0h | Reserved |
| 14-10 | SEL_2 | R/W | 0h | LUT4 block IN0 select inputs for unit 2. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |
| 9-5 | SEL_1 | R/W | 0h | LUT4 block IN0 select inputs for unit 1. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |
| 4-0 | SEL_0 | R/W | 0h | LUT4 block IN0 select inputs for unit 0. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.2.10 CLB_LUT4_IN1 Register (Offset = 14h) [Reset = 00000000h]

CLB_LUT4_IN1 is shown in Figure 26-25 and described in Table 26-26.

Return to the Summary Table.

LUT4_0/1/2 IN1 input source

**Figure 26-25. CLB_LUT4_IN1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | | | SEL_2 | | | | | SEL_1 | | | | | SEL_0 | | | | |
| R/W1C-0h | | | | | | | | | | | | | | | | | R/W-0h | | | | | R/W-0h | | | | | R/W-0h | | | | |

**Table 26-26. CLB_LUT4_IN1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-15 | RESERVED | R/W1C | 0h | Reserved |
| 14-10 | SEL_2 | R/W | 0h | LUT4 block IN1 select inputs for unit 2. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |
| 9-5 | SEL_1 | R/W | 0h | LUT4 block IN1 select inputs for unit 1. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |
| 4-0 | SEL_0 | R/W | 0h | LUT4 block IN1 select inputs for unit 0. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.2.11 CLB_LUT4_IN2 Register (Offset = 16h) [Reset = 00000000h]

CLB_LUT4_IN2 is shown in Figure 26-26 and described in Table 26-27.

Return to the Summary Table.

LUT4_0/1/2 IN2 input source

**Figure 26-26. CLB_LUT4_IN2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | | | SEL_2 | | | | | SEL_1 | | | | | SEL_0 | | | | |
| R/W1C-0h | | | | | | | | | | | | | | | | | R/W-0h | | | | | R/W-0h | | | | | R/W-0h | | | | |

**Table 26-27. CLB_LUT4_IN2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-15 | RESERVED | R/W1C | 0h | Reserved |
| 14-10 | SEL_2 | R/W | 0h | LUT4 block IN2 select inputs for unit 2. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |
| 9-5 | SEL_1 | R/W | 0h | LUT4 block IN2 select inputs for unit 1. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |
| 4-0 | SEL_0 | R/W | 0h | LUT4 block IN2 select inputs for unit 0. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.2.12 CLB_LUT4_IN3 Register (Offset = 18h) [Reset = 00000000h]

CLB_LUT4_IN3 is shown in Figure 26-27 and described in Table 26-28.

Return to the Summary Table.

LUT4_0/1/2 IN3 input source

**Figure 26-27. CLB_LUT4_IN3 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | | | SEL_2 | | | | | SEL_1 | | | | | SEL_0 | | | | |
| R/W1C-0h | | | | | | | | | | | | | | | | | R/W-0h | | | | | R/W-0h | | | | | R/W-0h | | | | |

**Table 26-28. CLB_LUT4_IN3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-15 | RESERVED | R/W1C | 0h | Reserved |
| 14-10 | SEL_2 | R/W | 0h | LUT4 block IN3 select inputs for unit 2. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |
| 9-5 | SEL_1 | R/W | 0h | LUT4 block IN3 select inputs for unit 1. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |
| 4-0 | SEL_0 | R/W | 0h | LUT4 block IN3 select inputs for unit 0. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.2.13 CLB_FSM_LUT_FN1_0 Register (Offset = 1Ch) [Reset = 00000000h]

CLB_FSM_LUT_FN1_0 is shown in Figure 26-28 and described in Table 26-29.

Return to the Summary Table.

LUT function for FSM Unit 1 and Unit 0

**Figure 26-28. CLB_FSM_LUT_FN1_0 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FN1 | | | | | | | | | | | | | | | | FN0 | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | | R/W-0h | | | | | | | | | | | | | | | |

**Table 26-29. CLB_FSM_LUT_FN1_0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | FN1 | R/W | 0h | FSM block LUT output function for unit 1<br>Reset type: SYSRSn |
| 15-0 | FN0 | R/W | 0h | FSM block LUT output function for unit 0<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.2.14 CLB_FSM_LUT_FN2 Register (Offset = 1Eh) [Reset = 00000000h]

CLB_FSM_LUT_FN2 is shown in Figure 26-29 and described in Table 26-30.

Return to the Summary Table.

LUT function for FSM Unit 2

**Figure 26-29. CLB_FSM_LUT_FN2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | | FN1 | | | | | | | | | | | | | | | |
| R/W1C-0h | | | | | | | | | | | | | | | | R/W-0h | | | | | | | | | | | | | | | |

**Table 26-30. CLB_FSM_LUT_FN2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R/W1C | 0h | Reserved |
| 15-0 | FN1 | R/W | 0h | LUT4 output function for unit 2<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.2.15 CLB_LUT4_FN1_0 Register (Offset = 20h) [Reset = 00000000h]

CLB_LUT4_FN1_0 is shown in Figure 26-30 and described in Table 26-31.

Return to the Summary Table.

LUT function for LUT4 block of Unit 1 and 0

**Figure 26-30. CLB_LUT4_FN1_0 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FN1 | | | | | | | | | | | | | | | | FN0 | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | | R/W-0h | | | | | | | | | | | | | | | |

**Table 26-31. CLB_LUT4_FN1_0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | FN1 | R/W | 0h | LUT4 output function for unit 1<br>Reset type: SYSRSn |
| 15-0 | FN0 | R/W | 0h | LUT4 output function for unit 0<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.2.16 CLB_LUT4_FN2 Register (Offset = 22h) [Reset = 00000000h]

CLB_LUT4_FN2 is shown in Figure 26-31 and described in Table 26-32.

Return to the Summary Table.

LUT function for LUT4 block of Unit 2

**Figure 26-31. CLB_LUT4_FN2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | | FN1 | | | | | | | | | | | | | | | |
| R/W1C-0h | | | | | | | | | | | | | | | | R/W-0h | | | | | | | | | | | | | | | |

**Table 26-32. CLB_LUT4_FN2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R/W1C | 0h | Reserved |
| 15-0 | FN1 | R/W | 0h | LUT4 output function for unit 2<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.2.17 CLB_FSM_NEXT_STATE_0 Register (Offset = 24h) [Reset = 00000000h]

CLB_FSM_NEXT_STATE_0 is shown in Figure 26-32 and described in Table 26-33.

Return to the Summary Table.

FSM Next state equations for Unit 0

**Figure 26-32. CLB_FSM_NEXT_STATE_0 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S1 | | | | | | | | | | | | | | | | S0 | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | | R/W-0h | | | | | | | | | | | | | | | |

**Table 26-33. CLB_FSM_NEXT_STATE_0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | S1 | R/W | 0h | FSM next state function for S1, unit0<br>Reset type: SYSRSn |
| 15-0 | S0 | R/W | 0h | FSM next state function for S0, unit0<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.2.18 CLB_FSM_NEXT_STATE_1 Register (Offset = 26h) [Reset = 00000000h]

CLB_FSM_NEXT_STATE_1 is shown in Figure 26-33 and described in Table 26-34.

Return to the Summary Table.

FSM Next state equations for Unit 1

**Figure 26-33. CLB_FSM_NEXT_STATE_1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S1 | | | | | | | | | | | | | | | | S0 | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | | R/W-0h | | | | | | | | | | | | | | | |

**Table 26-34. CLB_FSM_NEXT_STATE_1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | S1 | R/W | 0h | FSM next state function for S1, unit1<br>Reset type: SYSRSn |
| 15-0 | S0 | R/W | 0h | FSM next state function for S0, unit1<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.2.19 CLB_FSM_NEXT_STATE_2 Register (Offset = 28h) [Reset = 00000000h]

CLB_FSM_NEXT_STATE_2 is shown in Figure 26-34 and described in Table 26-35.

Return to the Summary Table.

FSM Next state equations for Unit 2

**Figure 26-34. CLB_FSM_NEXT_STATE_2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S1 | | | | | | | | | | | | | | | | S0 | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | | R/W-0h | | | | | | | | | | | | | | | |

**Table 26-35. CLB_FSM_NEXT_STATE_2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | S1 | R/W | 0h | FSM next state function for S1, unit2<br>Reset type: SYSRSn |
| 15-0 | S0 | R/W | 0h | FSM next state function for S0, unit2<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.2.20 CLB_MISC_CONTROL Register (Offset = 2Ah) [Reset = 00000000h]

CLB_MISC_CONTROL is shown in Figure 26-35 and described in Table 26-36.

Return to the Summary Table.

Static controls for Ctr,FSM

**Figure 26-35. CLB_MISC_CONTROL Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | FSM_EXTRA_SEL1_2 | FSM_EXTRA_SEL0_2 |
| R-0-0h | | | | | | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| FSM_EXTRA_SEL1_1 | FSM_EXTRA_SEL0_1 | FSM_EXTRA_SEL1_0 | FSM_EXTRA_SEL0_0 | RESERVED | | | COUNT_EVENT_CTRL_2 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R-0-0h | | | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| COUNT_DIR_2 | COUNT_ADD_SHIFT_2 | COUNT_EVENT_CTRL_1 | COUNT_DIR_1 | COUNT_ADD_SHIFT_1 | COUNT_EVENT_CTRL_0 | COUNT_DIR_0 | COUNT_ADD_SHIFT_0 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 26-36. CLB_MISC_CONTROL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-18 | RESERVED | R-0 | 0h | Reserved |
| 17 | FSM_EXTRA_SEL1_2 | R/W | 0h | Defines which input should be selected for the FSM LUT of UNIT 2<br>0 = Selects State S1 for the FSM LUT<br>1 = Selects EXTRA_EXT_IN1 for the FSM LUT<br>Reset type: SYSRSn |
| 16 | FSM_EXTRA_SEL0_2 | R/W | 0h | Defines which input should be selected for the FSM LUT of UNIT 2<br>0 = Selects State S0 for the FSM LUT<br>1 = Selects EXTRA_EXT_IN0 for the FSM LUT<br>Reset type: SYSRSn |
| 15 | FSM_EXTRA_SEL1_1 | R/W | 0h | Defines which input should be selected for the FSM LUT of UNIT 1<br>0 = Selects State S1 for the FSM LUT<br>1 = Selects EXTRA_EXT_IN1 for the FSM LUT<br>Reset type: SYSRSn |
| 14 | FSM_EXTRA_SEL0_1 | R/W | 0h | Defines which input should be selected for the FSM LUT of UNIT 1<br>0 = Selects State S0 for the FSM LUT<br>1 = Selects EXTRA_EXT_IN0 for the FSM LUT<br>Reset type: SYSRSn |
| 13 | FSM_EXTRA_SEL1_0 | R/W | 0h | Defines which input should be selected for the FSM LUT of UNIT 0<br>0 = Selects State S1 for the FSM LUT<br>1 = Selects EXTRA_EXT_IN1 for the FSM LUT<br>Reset type: SYSRSn |
| 12 | FSM_EXTRA_SEL0_0 | R/W | 0h | Defines which input should be selected for the FSM LUT of UNIT 0<br>0 = Selects State S0 for the FSM LUT<br>1 = Selects EXTRA_EXT_IN0 for the FSM LUT<br>Reset type: SYSRSn |
| 11-9 | RESERVED | R-0 | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

**Table 26-36. CLB_MISC_CONTROL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 8 | COUNT_EVENT_CTRL_2 | R/W | 0h | Controls the actions on an EVENT for UNIT2. Must be 0 for indirect loads and HLC loads of the counter to take effect.<br>0 = No add or shift, but load the predefined value<br>1 = Based on other bits, add/shift with the predefined value<br>Reset type: SYSRSn |
| 7 | COUNT_DIR_2 | R/W | 0h | Controls add/shift direction for UNIT 2<br>0 = right shift or subtract<br>1 = left shift or add<br>Reset type: SYSRSn |
| 6 | COUNT_ADD_SHIFT_2 | R/W | 0h | Controls whether the UNIT 2 counter will do an ADD or a SHIFT on an EVENT.<br>0 =Shift<br>1 = ADD<br>Reset type: SYSRSn |
| 5 | COUNT_EVENT_CTRL_1 | R/W | 0h | Controls the actions on an EVENT for UNIT1. Must be 0 for indirect loads and HLC loads of the counter to take effect.<br>0 = No add or shift, but load the predefined value<br>1 = Based on other bits, add/shift with the predefined value<br>Reset type: SYSRSn |
| 4 | COUNT_DIR_1 | R/W | 0h | Controls add/shift direction for UNIT 1<br>0 = right shift or subtract<br>1 = left shift or add<br>Reset type: SYSRSn |
| 3 | COUNT_ADD_SHIFT_1 | R/W | 0h | Controls whether the UNIT 1 counter will do an ADD or a SHIFT on an EVENT.<br>0 = Shift<br>1 = ADD<br>Reset type: SYSRSn |
| 2 | COUNT_EVENT_CTRL_0 | R/W | 0h | Controls the actions on an EVENT for UNIT1. Must be 0 for indirect loads and HLC loads of the counter to take effect.<br>0 = No add or shift, but load the predefined value<br>1 = Based on other bits, add/shift with the predefined value<br>Reset type: SYSRSn |
| 1 | COUNT_DIR_0 | R/W | 0h | Controls add/shift direction for UNIT 0<br>0 = right shift or subtract<br>1 = left shift or add<br>Reset type: SYSRSn |
| 0 | COUNT_ADD_SHIFT_0 | R/W | 0h | Controls whether the UNIT 0 counter will do an ADD or a SHIFT on an EVENT.<br>0 = Shift<br>1 = ADD<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.2.21 CLB_OUTPUT_LUT_0 Register (Offset = 2Ch) [Reset = 00000000h]

CLB_OUTPUT_LUT_0 is shown in Figure 26-36 and described in Table 26-37.

Return to the Summary Table.

Inp Sel, LUT fns for Out0

**Figure 26-36. CLB_OUTPUT_LUT_0 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | FN | | | | | | | | IN2 | | | | | IN1 | | | | | IN0 | | | | |
| R/W1C-0h | | | | | | | | | R/W-0h | | | | | | | | R/W-0h | | | | | R/W-0h | | | | | R/W-0h | | | | |

**Table 26-37. CLB_OUTPUT_LUT_0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-23 | RESERVED | R/W1C | 0h | Reserved |
| 22-15 | FN | R/W | 0h | Output function for output LUT<br>Reset type: SYSRSn |
| 14-10 | IN2 | R/W | 0h | Select value for IN2 of output LUT. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |
| 9-5 | IN1 | R/W | 0h | Select value for IN1 of output LUT. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |
| 4-0 | IN0 | R/W | 0h | Select value for IN0 of output LUT. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.2.22 CLB_OUTPUT_LUT_1 Register (Offset = 2Eh) [Reset = 00000000h]

CLB_OUTPUT_LUT_1 is shown in Figure 26-37 and described in Table 26-38.

Return to the Summary Table.

Inp Sel, LUT fns for Out1

**Figure 26-37. CLB_OUTPUT_LUT_1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | FN | | | | | | | | IN2 | | | | | IN1 | | | | | IN0 | | | | |
| R/W1C-0h | | | | | | | | | R/W-0h | | | | | | | | R/W-0h | | | | | R/W-0h | | | | | R/W-0h | | | | |

**Table 26-38. CLB_OUTPUT_LUT_1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-23 | RESERVED | R/W1C | 0h | Reserved |
| 22-15 | FN | R/W | 0h | Output function for output LUT<br>Reset type: SYSRSn |
| 14-10 | IN2 | R/W | 0h | Select value for IN2 of output LUT. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |
| 9-5 | IN1 | R/W | 0h | Select value for IN1 of output LUT. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |
| 4-0 | IN0 | R/W | 0h | Select value for IN0 of output LUT. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.2.23 CLB_OUTPUT_LUT_2 Register (Offset = 30h) [Reset = 00000000h]

CLB_OUTPUT_LUT_2 is shown in Figure 26-38 and described in Table 26-39.

Return to the Summary Table.

Inp Sel, LUT fns for Out2

**Figure 26-38. CLB_OUTPUT_LUT_2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | FN | | | | | | | | IN2 | | | | | IN1 | | | | | IN0 | | | | |
| R/W1C-0h | | | | | | | | | R/W-0h | | | | | | | | R/W-0h | | | | | R/W-0h | | | | | R/W-0h | | | | |

**Table 26-39. CLB_OUTPUT_LUT_2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-23 | RESERVED | R/W1C | 0h | Reserved |
| 22-15 | FN | R/W | 0h | Output function for output LUT<br>Reset type: SYSRSn |
| 14-10 | IN2 | R/W | 0h | Select value for IN2 of output LUT. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |
| 9-5 | IN1 | R/W | 0h | Select value for IN1 of output LUT. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |
| 4-0 | IN0 | R/W | 0h | Select value for IN0 of output LUT. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.2.24 CLB_OUTPUT_LUT_3 Register (Offset = 32h) [Reset = 00000000h]

CLB_OUTPUT_LUT_3 is shown in Figure 26-39 and described in Table 26-40.

Return to the Summary Table.

Inp Sel, LUT fns for Out3

**Figure 26-39. CLB_OUTPUT_LUT_3 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | FN | | | | | | | | IN2 | | | | | IN1 | | | | | IN0 | | | | |
| R/W1C-0h | | | | | | | | | R/W-0h | | | | | | | | R/W-0h | | | | | R/W-0h | | | | | R/W-0h | | | | |

**Table 26-40. CLB_OUTPUT_LUT_3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-23 | RESERVED | R/W1C | 0h | Reserved |
| 22-15 | FN | R/W | 0h | Output function for output LUT<br>Reset type: SYSRSn |
| 14-10 | IN2 | R/W | 0h | Select value for IN2 of output LUT. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |
| 9-5 | IN1 | R/W | 0h | Select value for IN1 of output LUT. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |
| 4-0 | IN0 | R/W | 0h | Select value for IN0 of output LUT. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.2.25 CLB_OUTPUT_LUT_4 Register (Offset = 34h) [Reset = 00000000h]

CLB_OUTPUT_LUT_4 is shown in Figure 26-40 and described in Table 26-41.

Return to the Summary Table.

Inp Sel, LUT fns for Out4

**Figure 26-40. CLB_OUTPUT_LUT_4 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | FN | | | | | | | | IN2 | | | | | IN1 | | | | | IN0 | | | | |
| R/W1C-0h | | | | | | | | | R/W-0h | | | | | | | | R/W-0h | | | | | R/W-0h | | | | | R/W-0h | | | | |

**Table 26-41. CLB_OUTPUT_LUT_4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-23 | RESERVED | R/W1C | 0h | Reserved |
| 22-15 | FN | R/W | 0h | Output function for output LUT<br>Reset type: SYSRSn |
| 14-10 | IN2 | R/W | 0h | Select value for IN2 of output LUT. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |
| 9-5 | IN1 | R/W | 0h | Select value for IN1 of output LUT. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |
| 4-0 | IN0 | R/W | 0h | Select value for IN0 of output LUT. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.2.26 CLB_OUTPUT_LUT_5 Register (Offset = 36h) [Reset = 00000000h]

CLB_OUTPUT_LUT_5 is shown in Figure 26-41 and described in Table 26-42.

Return to the Summary Table.

Inp Sel, LUT fns for Out5

**Figure 26-41. CLB_OUTPUT_LUT_5 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | FN | | | | | | | | IN2 | | | | | IN1 | | | | | IN0 | | | | |
| R/W1C-0h | | | | | | | | | R/W-0h | | | | | | | | R/W-0h | | | | | R/W-0h | | | | | R/W-0h | | | | |

**Table 26-42. CLB_OUTPUT_LUT_5 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-23 | RESERVED | R/W1C | 0h | Reserved |
| 22-15 | FN | R/W | 0h | Output function for output LUT<br>Reset type: SYSRSn |
| 14-10 | IN2 | R/W | 0h | Select value for IN2 of output LUT. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |
| 9-5 | IN1 | R/W | 0h | Select value for IN1 of output LUT. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |
| 4-0 | IN0 | R/W | 0h | Select value for IN0 of output LUT. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.2.27 CLB_OUTPUT_LUT_6 Register (Offset = 38h) [Reset = 00000000h]

CLB_OUTPUT_LUT_6 is shown in Figure 26-42 and described in Table 26-43.

Return to the Summary Table.

Inp Sel, LUT fns for Out6

**Figure 26-42. CLB_OUTPUT_LUT_6 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | FN | | | | | | | | IN2 | | | | | IN1 | | | | | IN0 | | | | |
| R/W1C-0h | | | | | | | | | R/W-0h | | | | | | | | R/W-0h | | | | | R/W-0h | | | | | R/W-0h | | | | |

**Table 26-43. CLB_OUTPUT_LUT_6 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-23 | RESERVED | R/W1C | 0h | Reserved |
| 22-15 | FN | R/W | 0h | Output function for output LUT<br>Reset type: SYSRSn |
| 14-10 | IN2 | R/W | 0h | Select value for IN2 of output LUT. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |
| 9-5 | IN1 | R/W | 0h | Select value for IN1 of output LUT. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |
| 4-0 | IN0 | R/W | 0h | Select value for IN0 of output LUT. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.2.28 CLB_OUTPUT_LUT_7 Register (Offset = 3Ah) [Reset = 00000000h]

CLB_OUTPUT_LUT_7 is shown in Figure 26-43 and described in Table 26-44.

Return to the Summary Table.

Inp Sel, LUT fns for Out7

**Figure 26-43. CLB_OUTPUT_LUT_7 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | FN | | | | | | | | IN2 | | | | | IN1 | | | | | IN0 | | | | |
| R/W1C-0h | | | | | | | | | R/W-0h | | | | | | | | R/W-0h | | | | | R/W-0h | | | | | R/W-0h | | | | |

**Table 26-44. CLB_OUTPUT_LUT_7 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-23 | RESERVED | R/W1C | 0h | Reserved |
| 22-15 | FN | R/W | 0h | Output function for output LUT<br>Reset type: SYSRSn |
| 14-10 | IN2 | R/W | 0h | Select value for IN2 of output LUT. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |
| 9-5 | IN1 | R/W | 0h | Select value for IN1 of output LUT. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |
| 4-0 | IN0 | R/W | 0h | Select value for IN0 of output LUT. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.2.29 CLB_HLC_EVENT_SEL Register (Offset = 3Ch) [Reset = 00000000h]

CLB_HLC_EVENT_SEL is shown in Figure 26-44 and described in Table 26-45.

Return to the Summary Table.

Event Selector register for the High Level controller

**Figure 26-44. CLB_HLC_EVENT_SEL Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | EVENT3_SEL | | | |
| R-0-0h | | | | | | | | | | | | R/W-0h | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EVENT3_SEL | EVENT2_SEL | | | | | EVENT1_SEL | | | | | EVENT0_SEL | | | | |
| R/W-0h | R/W-0h | | | | | R/W-0h | | | | | R/W-0h | | | | |

**Table 26-45. CLB_HLC_EVENT_SEL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-20 | RESERVED | R-0 | 0h | Reserved |
| 19-15 | EVENT3_SEL | R/W | 0h | 5 bit select value for EVENT3 of the High Level Controller. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |
| 14-10 | EVENT2_SEL | R/W | 0h | 5 bit select value for EVENT2 of the High Level Controller. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |
| 9-5 | EVENT1_SEL | R/W | 0h | 5 bit select value for EVENT1 of the High Level Controller. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |
| 4-0 | EVENT0_SEL | R/W | 0h | 5 bit select value for EVENT0 of the High Level Controller. See the Static Switch Block Output Mux Table.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.3 CLB_LOGIC_CONTROL_REGS Registers

Table 26-46 lists the memory-mapped registers for the CLB_LOGIC_CONTROL_REGS registers. All register offset addresses not listed in Table 26-46 should be considered as reserved locations and the register contents should not be modified.

**Table 26-46. CLB_LOGIC_CONTROL_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | CLB_LOAD_EN | Global enable & indirect load enable control, only Global Enable Bit is LOCK protected | LOCK | Go |
| 2h | CLB_LOAD_ADDR | Indirect address | | Go |
| 4h | CLB_LOAD_DATA | Data for indirect loads | | Go |
| 6h | CLB_INPUT_FILTER | Input filter selection for both edge detection and synchronizers | LOCK | Go |
| 8h | CLB_IN_MUX_SEL_0 | Input selection to decide between Signals and GP register | LOCK | Go |
| Ah | CLB_LCL_MUX_SEL_1 | Input Mux selection for local mux | LOCK | Go |
| Ch | CLB_LCL_MUX_SEL_2 | Input Mux selection for local mux | LOCK | Go |
| Eh | CLB_BUF_PTR | PUSH and PULL pointers | | Go |
| 10h | CLB_GP_REG | General purpose register for CELL inputs | | Go |
| 12h | CLB_OUT_EN | CELL output enable register | | Go |
| 14h | CLB_GLBL_MUX_SEL_1 | Global Mux select for CELL inputs | LOCK | Go |
| 16h | CLB_GLBL_MUX_SEL_2 | Global Mux select for CELL inputs | LOCK | Go |
| 20h | CLB_INTR_TAG_REG | Interrupt Tag register | | Go |
| 22h | CLB_LOCK | Lock control register | EALLOW | Go |
| 30h | CLB_DBG_R0 | R0 of High level Controller | | Go |
| 32h | CLB_DBG_R1 | R1 of High level Controller | | Go |
| 34h | CLB_DBG_R2 | R2 of High level Controller | | Go |
| 36h | CLB_DBG_R3 | R3 of High level Controller | | Go |
| 38h | CLB_DBG_C0 | Count of Unit 0 | | Go |
| 3Ah | CLB_DBG_C1 | Count of Unit 1 | | Go |
| 3Ch | CLB_DBG_C2 | Count of Unit 2 | | Go |
| 3Eh | CLB_DBG_OUT | Outputs of various units in the Cell | | Go |

Complex bit access types are encoded to fit into small table cells. Table 26-47 shows the codes that are used for access types in this section.

**Table 26-47. CLB_LOGIC_CONTROL_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| Read Type | | |
| R | R | Read |
| R-0 | R<br>-0 | Read<br>Returns 0s |
| R-1 | R<br>-1 | Read<br>Returns 1s |
| Write Type | | |
| W | W | Write |
| WSonce | W<br>Sonce | Write<br>Set once |
| Reset or Default Value | | |
| *-n* | | Value after reset or the default value |



Copyright © 2024 Texas Instruments Incorporated

**Table 26-47. CLB_LOGIC_CONTROL_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. |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.3.1 CLB_LOAD_EN Register (Offset = 0h) [Reset = 0000h]

CLB_LOAD_EN is shown in Figure 26-45 and described in Table 26-48.

Return to the Summary Table.

Global enable & indirect load enable control, only Global Enable Bit is LOCK protected

**Figure 26-45. CLB_LOAD_EN Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | STOP | GLOBAL_EN | LOAD_EN |
| R-0-0h | | | | | R/W-0h | R/W-0h | R/W-0h |

**Table 26-48. CLB_LOAD_EN Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-3 | RESERVED | R-0 | 0h | Reserved |
| 2 | STOP | R/W | 0h | This bit defines the behaviour of the sequential elements in the CELL during debug HALTs of the CPU. If this bit is set to 0, the debug HALT condition is ignored.<br>Reset type: SYSRSn |
| 1 | GLOBAL_EN | R/W | 0h | This bit is a global enable signal for the logic in the CELL. This also acts as a soft reset for the CELL logic. CLB outputs (including LUTs and OUTLUTs) will be gated when this bit is cleared from 1 to 0, i.e., the CLB outputs will be low when GLOBAL_EN is low. Additionally, the FSM and AOC blocks will also be reset. Note that when this bit goes low, the COUNTER blocks and HLC are simply halted, but they will NOT be reset internally. This allows the ability to preload these submodules when GLOBAL_EN is 0. This bit is normally set after all the other configuration settings are completed. This bit is LOCK protected.<br>Reset type: SYSRSn |
| 0 | LOAD_EN | R/W | 0h | A write with this bit set to 1 will pulse the Load Enable signal for the indirect register loads in the CELL.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.3.2 CLB_LOAD_ADDR Register (Offset = 2h) [Reset = 00000000h]

CLB_LOAD_ADDR is shown in Figure 26-46 and described in Table 26-49.

Return to the Summary Table.

Indirect address

**Figure 26-46. CLB_LOAD_ADDR Register**

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

**Table 26-49. CLB_LOAD_ADDR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R-0 | 0h | Reserved |
| 5-0 | ADDR | R/W | 0h | These are the address bits used for writing to the indirect address space of the CELL.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.3.3 CLB_LOAD_DATA Register (Offset = 4h) [Reset = 00000000h]

CLB_LOAD_DATA is shown in Figure 26-47 and described in Table 26-50.

Return to the Summary Table.

Data for indirect loads

**Figure 26-47. CLB_LOAD_DATA Register**

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

**Table 26-50. CLB_LOAD_DATA Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | DATA | R/W | 0h | This register holds the 32-bit data for writing to the indirect address space of the CELL.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.3.4 CLB_INPUT_FILTER Register (Offset = 6h) [Reset = 00000000h]

CLB_INPUT_FILTER is shown in Figure 26-48 and described in Table 26-51.

Return to the Summary Table.

Input filter selection for both edge detection and synchronizers

**Figure 26-48. CLB_INPUT_FILTER Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| SYNC7 | SYNC6 | SYNC5 | SYNC4 | SYNC3 | SYNC2 | SYNC1 | SYNC0 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| FIN7 | | FIN6 | | FIN5 | | FIN4 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| FIN3 | | FIN2 | | FIN1 | | FIN0 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 26-51. CLB_INPUT_FILTER Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-24 | RESERVED | R-0 | 0h | Reserved |
| 23 | SYNC7 | R/W | 0h | Synchronizer Select Control for Input 7<br>Reset type: SYSRSn |
| 22 | SYNC6 | R/W | 0h | Synchronizer Select Control for Input 6<br>Reset type: SYSRSn |
| 21 | SYNC5 | R/W | 0h | Synchronizer Select Control for Input 5<br>Reset type: SYSRSn |
| 20 | SYNC4 | R/W | 0h | Synchronizer Select Control for Input 4<br>Reset type: SYSRSn |
| 19 | SYNC3 | R/W | 0h | Synchronizer Select Control for Input 3<br>Reset type: SYSRSn |
| 18 | SYNC2 | R/W | 0h | Synchronizer Select Control for Input 2<br>Reset type: SYSRSn |
| 17 | SYNC1 | R/W | 0h | Synchronizer Select Control for Input 1<br>Reset type: SYSRSn |
| 16 | SYNC0 | R/W | 0h | Synchronizer Select Control for Input 0<br>Reset type: SYSRSn |
| 15-14 | FIN7 | R/W | 0h | Input filter selection for CELL Input 7<br>2 bits are used to define the edge filtering .<br>00 : No filtering<br>01 : Rising edge detect<br>10 : Falling edge detect<br>11 : Any edge detect<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 26-51. CLB_INPUT_FILTER Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 13-12 | FIN6 | R/W | 0h | Input filter selection for CELL Input 6<br>2 bits are used to define the edge filtering .<br>00 : No filtering<br>01 : Rising edge detect<br>10 : Falling edge detect<br>11 : Any edge detect<br>Reset type: SYSRSn |
| 11-10 | FIN5 | R/W | 0h | Input filter selection for CELL Input 5<br>2 bits are used to define the edge filtering .<br>00 : No filtering<br>01 : Rising edge detect<br>10 : Falling edge detect<br>11 : Any edge detect<br>Reset type: SYSRSn |
| 9-8 | FIN4 | R/W | 0h | Input filter selection for CELL Input 4<br>2 bits are used to define the edge filtering .<br>00 : No filtering<br>01 : Rising edge detect<br>10 : Falling edge detect<br>11 : Any edge detect<br>Reset type: SYSRSn |
| 7-6 | FIN3 | R/W | 0h | Input filter selection for CELL Input 3<br>2 bits are used to define the edge filtering .<br>00 : No filtering<br>01 : Rising edge detect<br>10 : Falling edge detect<br>11 : Any edge detect<br>Reset type: SYSRSn |
| 5-4 | FIN2 | R/W | 0h | Input filter selection for CELL Input 2<br>2 bits are used to define the edge filtering .<br>00 : No filtering<br>01 : Rising edge detect<br>10 : Falling edge detect<br>11 : Any edge detect<br>Reset type: SYSRSn |
| 3-2 | FIN1 | R/W | 0h | Input filter selection for CELL Input 1<br>2 bits are used to define the edge filtering .<br>00 : No filtering<br>01 : Rising edge detect<br>10 : Falling edge detect<br>11 : Any edge detect<br>Reset type: SYSRSn |
| 1-0 | FIN0 | R/W | 0h | Input filter selection for CELL Input 0<br>2 bits are used to define the edge filtering .<br>00 : No filtering<br>01 : Rising edge detect<br>10 : Falling edge detect<br>11 : Any edge detect<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.3.5 CLB_IN_MUX_SEL_0 Register (Offset = 8h) [Reset = 00000000h]

CLB_IN_MUX_SEL_0 is shown in Figure 26-49 and described in Table 26-52.

Return to the Summary Table.

Input selection to decide between Signals and GP register

**Figure 26-49. CLB_IN_MUX_SEL_0 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SEL_GP_IN_7 | SEL_GP_IN_6 | SEL_GP_IN_5 | SEL_GP_IN_4 | SEL_GP_IN_3 | SEL_GP_IN_2 | SEL_GP_IN_1 | SEL_GP_IN_0 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 26-52. CLB_IN_MUX_SEL_0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-8 | RESERVED | R-0 | 0h | Reserved |
| 7 | SEL_GP_IN_7 | R/W | 0h | Select control for Input 7 to decide between external input and CLB_GP_REG[7]<br>0 : Input comes from selected external input<br>1 : Input comes from CLB_GP_REG[7]<br>Reset type: SYSRSn |
| 6 | SEL_GP_IN_6 | R/W | 0h | Select control for Input 6 to decide between external input and CLB_GP_REG[6]<br>0 : Input comes from selected external input<br>1 : Input comes from CLB_GP_REG[6]<br>Reset type: SYSRSn |
| 5 | SEL_GP_IN_5 | R/W | 0h | Select control for Input 5 to decide between external input and CLB_GP_REG[5]<br>0 : Input comes from selected external input<br>1 : Input comes from CLB_GP_REG[5]<br>Reset type: SYSRSn |
| 4 | SEL_GP_IN_4 | R/W | 0h | Select control for Input 4 to decide between external input and CLB_GP_REG[4]<br>0 : Input comes from selected external input<br>1 : Input comes from CLB_GP_REG[4]<br>Reset type: SYSRSn |
| 3 | SEL_GP_IN_3 | R/W | 0h | Select control for Input 3 to decide between external input and CLB_GP_REG[3]<br>0 : Input comes from selected external input<br>1 : Input comes from CLB_GP_REG[3]<br>Reset type: SYSRSn |
| 2 | SEL_GP_IN_2 | R/W | 0h | Select control for Input 2 to decide between external input and CLB_GP_REG[2]<br>0 : Input comes from selected external input<br>1 : Input comes from CLB_GP_REG[2]<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 26-52. CLB_IN_MUX_SEL_0 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1 | SEL_GP_IN_1 | R/W | 0h | Select control for Input 1 to decide between external input and CLB_GP_REG[1]<br>0 : Input comes from selected external input<br>1 : Input comes from CLB_GP_REG[1]<br>Reset type: SYSRSn |
| 0 | SEL_GP_IN_0 | R/W | 0h | Select control for Input 0 to decide between external input and CLB_GP_REG[0]<br>0 : Input comes from selected external input<br>1 : Input comes from CLB_GP_REG[0]<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.3.6 CLB_LCL_MUX_SEL_1 Register (Offset = Ah) [Reset = 00000000h]

CLB_LCL_MUX_SEL_1 is shown in Figure 26-50 and described in Table 26-53.

Return to the Summary Table.

Input Mux selection for local mux

**Figure 26-50. CLB_LCL_MUX_SEL_1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | LCL_MUX_SEL_IN_3 | | | |
| R-0-0h | | | | R/W-0h | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| LCL_MUX_SEL_IN_3 | LCL_MUX_SEL_IN_2 | | | | | LCL_MUX_SEL_IN_1 | |
| R/W-0h | R/W-0h | | | | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| LCL_MUX_SEL_IN_1 | | | LCL_MUX_SEL_IN_0 | | | | |
| R/W-0h | | | R/W-0h | | | | |

**Table 26-53. CLB_LCL_MUX_SEL_1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-20 | RESERVED | R-0 | 0h | Reserved |
| 19-15 | LCL_MUX_SEL_IN_3 | R/W | 0h | 5 bit MUX Select for Local MUX control for Input 3<br>See Local Signals and Mux Selection Table<br>Reset type: SYSRSn |
| 14-10 | LCL_MUX_SEL_IN_2 | R/W | 0h | 5 bit MUX Select for Local MUX control for Input 2<br>See Local Signals and Mux Selection Table<br>Reset type: SYSRSn |
| 9-5 | LCL_MUX_SEL_IN_1 | R/W | 0h | 5 bit MUX Select for Local MUX control for Input 1<br>See Local Signals and Mux Selection Table<br>Reset type: SYSRSn |
| 4-0 | LCL_MUX_SEL_IN_0 | R/W | 0h | 5 bit MUX Select for Local MUX control for Input 0<br>See Local Signals and Mux Selection Table<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.3.7 CLB_LCL_MUX_SEL_2 Register (Offset = Ch) [Reset = 00000000h]

CLB_LCL_MUX_SEL_2 is shown in Figure 26-51 and described in Table 26-54.

Return to the Summary Table.

Input Mux selection for local mux

**Figure 26-51. CLB_LCL_MUX_SEL_2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | LCL_MUX_SEL_IN_7 | | | |
| R-0-0h | | | | R/W-0h | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| LCL_MUX_SEL_IN_7 | LCL_MUX_SEL_IN_6 | | | | | LCL_MUX_SEL_IN_5 | |
| R/W-0h | R/W-0h | | | | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| LCL_MUX_SEL_IN_5 | | | LCL_MUX_SEL_IN_4 | | | | |
| R/W-0h | | | R/W-0h | | | | |

**Table 26-54. CLB_LCL_MUX_SEL_2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-20 | RESERVED | R-0 | 0h | Reserved |
| 19-15 | LCL_MUX_SEL_IN_7 | R/W | 0h | 5 bit MUX Select for Local MUX control for Input 7<br>See Local Signals and Mux Selection Table<br>Reset type: SYSRSn |
| 14-10 | LCL_MUX_SEL_IN_6 | R/W | 0h | 5 bit MUX Select for Local MUX control for Input 6<br>See Local Signals and Mux Selection Table<br>Reset type: SYSRSn |
| 9-5 | LCL_MUX_SEL_IN_5 | R/W | 0h | 5 bit MUX Select for Local MUX control for Input 5<br>See Local Signals and Mux Selection Table<br>Reset type: SYSRSn |
| 4-0 | LCL_MUX_SEL_IN_4 | R/W | 0h | 5 bit MUX Select for Local MUX control for Input 4<br>See Local Signals and Mux Selection Table<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.3.8 CLB_BUF_PTR Register (Offset = Eh) [Reset = 00000000h]

CLB_BUF_PTR is shown in Figure 26-52 and described in Table 26-55.

Return to the Summary Table.

PUSH and PULL pointers

**Figure 26-52. CLB_BUF_PTR Register**

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

**Table 26-55. CLB_BUF_PTR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-24 | RESERVED | R-0 | 0h | Reserved |
| 23-16 | PUSH | R/W | 0h | 8 bit pointer which indicates the number of data values which have been pulled from the buffer by the High Level Controller. This counter will wrap around after 0xff. The Least significant 2 bits are used as the actual pointer for the operation.<br>Reset type: SYSRSn |
| 15-8 | RESERVED | R-0 | 0h | Reserved |
| 7-0 | PULL | R/W | 0h | 8 bit pointer which indicates the number of data values that have been written by the High Level controller into the buffer. The Least significant 2 bits are used as the actual pointer for the operation.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.3.9 CLB_GP_REG Register (Offset = 10h) [Reset = 00000000h]

CLB_GP_REG is shown in Figure 26-53 and described in Table 26-56.

Return to the Summary Table.

General purpose register for CELL inputs

**Figure 26-53. CLB_GP_REG Register**

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

**Table 26-56. CLB_GP_REG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-8 | RESERVED | R-0 | 0h | Reserved |
| 7-0 | REG | R/W | 0h | 8 bits which are directly connected to the 8 inputs of the CELL if that corresponding bit is selected in the CLB_IN_MUX_SEL_0 register<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.3.10 CLB_OUT_EN Register (Offset = 12h) [Reset = 00000000h]

CLB_OUT_EN is shown in Figure 26-54 and described in Table 26-57.

Return to the Summary Table.

CELL output enable register

**Figure 26-54. CLB_OUT_EN Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | | OUT0 | | | | | | | | | | | | | | | |
| R-0-0h | | | | | | | | | | | | | | | | R/W-0h | | | | | | | | | | | | | | | |

**Table 26-57. CLB_OUT_EN Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R-0 | 0h | Reserved |
| 15-0 | OUT0 | R/W | 0h | 16 bits which are directly driven out as OUTPUT_EN signals. Enabling bit x (x = 0:15) will override the corresponding peripheral signal muxed on the CLB OUTx.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.3.11 CLB_GLBL_MUX_SEL_1 Register (Offset = 14h) [Reset = 00000000h]

CLB_GLBL_MUX_SEL_1 is shown in Figure 26-55 and described in Table 26-58.

Return to the Summary Table.

Global Mux select for CELL inputs

**Figure 26-55. CLB_GLBL_MUX_SEL_1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | GLBL_MUX_SEL_IN_3 | | | | | | | GLBL_MUX_SEL_IN_2 | | | | |
| R-0-0h | | | | R/W-0h | | | | | | | R/W-0h | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GLBL_MUX_SEL_IN_2 | | | | GLBL_MUX_SEL_IN_1 | | | | | | | GLBL_MUX_SEL_IN_0 | | | | |
| R/W-0h | | | | R/W-0h | | | | | | | R/W-0h | | | | |

**Table 26-58. CLB_GLBL_MUX_SEL_1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-28 | RESERVED | R-0 | 0h | Reserved |
| 27-21 | GLBL_MUX_SEL_IN_3 | R/W | 0h | 7 bit MUX Select for Global MUX control for Input 3<br>See Global Signals and Mux Selection Table<br>Reset type: SYSRSn |
| 20-14 | GLBL_MUX_SEL_IN_2 | R/W | 0h | 7 bit MUX Select for Global MUX control for Input 2<br>See Global Signals and Mux Selection Table<br>Reset type: SYSRSn |
| 13-7 | GLBL_MUX_SEL_IN_1 | R/W | 0h | 7 bit MUX Select for Global MUX control for Input 1<br>See Global Signals and Mux Selection Table<br>Reset type: SYSRSn |
| 6-0 | GLBL_MUX_SEL_IN_0 | R/W | 0h | 7 bit MUX Select for Global MUX control for Input 0<br>See Global Signals and Mux Selection Table<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.3.12 CLB_GLBL_MUX_SEL_2 Register (Offset = 16h) [Reset = 00000000h]

CLB_GLBL_MUX_SEL_2 is shown in Figure 26-56 and described in Table 26-59.

Return to the Summary Table.

Global Mux select for CELL inputs

**Figure 26-56. CLB_GLBL_MUX_SEL_2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | GLBL_MUX_SEL_IN_7 | | | | | | | GLBL_MUX_SEL_IN_6 | | | | |
| R-0-0h | | | | R/W-0h | | | | | | | R/W-0h | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GLBL_MUX_SEL_IN_6 | | | | GLBL_MUX_SEL_IN_5 | | | | | | | GLBL_MUX_SEL_IN_4 | | | | |
| R/W-0h | | | | R/W-0h | | | | | | | R/W-0h | | | | |

**Table 26-59. CLB_GLBL_MUX_SEL_2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-28 | RESERVED | R-0 | 0h | Reserved |
| 27-21 | GLBL_MUX_SEL_IN_7 | R/W | 0h | 7 bit MUX Select for Global MUX control for Input 7<br>See Global Signals and Mux Selection Table<br>Reset type: SYSRSn |
| 20-14 | GLBL_MUX_SEL_IN_6 | R/W | 0h | 7 bit MUX Select for Global MUX control for Input 6<br>See Global Signals and Mux Selection Table<br>Reset type: SYSRSn |
| 13-7 | GLBL_MUX_SEL_IN_5 | R/W | 0h | 7 bit MUX Select for Global MUX control for Input 5<br>See Global Signals and Mux Selection Table<br>Reset type: SYSRSn |
| 6-0 | GLBL_MUX_SEL_IN_4 | R/W | 0h | 7 bit MUX Select for Global MUX control for Input 4<br>See Global Signals and Mux Selection Table<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.3.13 CLB_INTR_TAG_REG Register (Offset = 20h) [Reset = 0000h]

CLB_INTR_TAG_REG is shown in Figure 26-57 and described in Table 26-60.

Return to the Summary Table.

Interrupt Tag register

**Figure 26-57. CLB_INTR_TAG_REG Register**

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

**Table 26-60. CLB_INTR_TAG_REG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-6 | RESERVED | R-0 | 0h | Reserved |
| 5-0 | TAG | R/W | 0h | 6 bits which are used by the High Level Controller to set a tag value on flagging interrupts. This can be cleared through the VBUS interface since it is writeable through the VBUS.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.3.14 CLB_LOCK Register (Offset = 22h) [Reset = 00000000h]

CLB_LOCK is shown in Figure 26-58 and described in Table 26-61.

Return to the Summary Table.

Lock control register

**Figure 26-58. CLB_LOCK Register**

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

**Table 26-61. CLB_LOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | KEY | WSonce | 0h | These 16 bits act as a key to enable writes to Bit 0 of this register. The only time a '1' can be written to Bit 0 is by a single 32-bit write where bits 31:16 equal 0x5a5a and bit 0 is '1'. All other writes are ignored including separate 16-bit writes. This is EALLOW protected.<br>Reset type: SYSRSn |
| 15-1 | RESERVED | R-0 | 0h | Reserved |
| 0 | LOCK | R/W | 0h | This bit is used as a one-time write bit (Set Once). Once it is set to '1', only a reset (SYSRSN 0) will clear this bit back to 0.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.3.15 CLB_DBG_R0 Register (Offset = 30h) [Reset = 00000000h]

CLB_DBG_R0 is shown in Figure 26-59 and described in Table 26-62.

Return to the Summary Table.

R0 of High level Controller

**Figure 26-59. CLB_DBG_R0 Register**

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

**Table 26-62. CLB_DBG_R0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | DBG | R | 0h | CLB_DBG_R0<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.3.16 CLB_DBG_R1 Register (Offset = 32h) [Reset = 00000000h]

CLB_DBG_R1 is shown in Figure 26-60 and described in Table 26-63.

Return to the Summary Table.

R1 of High level Controller

**Figure 26-60. CLB_DBG_R1 Register**

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

**Table 26-63. CLB_DBG_R1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | DBG | R | 0h | CLB_DBG_R1<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.3.17 CLB_DBG_R2 Register (Offset = 34h) [Reset = 00000000h]

CLB_DBG_R2 is shown in Figure 26-61 and described in Table 26-64.

Return to the Summary Table.

R2 of High level Controller

**Figure 26-61. CLB_DBG_R2 Register**

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

**Table 26-64. CLB_DBG_R2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | DBG | R | 0h | CLB_DBG_R2<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.3.18 CLB_DBG_R3 Register (Offset = 36h) [Reset = 00000000h]

CLB_DBG_R3 is shown in Figure 26-62 and described in Table 26-65.

Return to the Summary Table.

R3 of High level Controller

**Figure 26-62. CLB_DBG_R3 Register**

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

**Table 26-65. CLB_DBG_R3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | DBG | R | 0h | CLB_DBG_R3<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.3.19 CLB_DBG_C0 Register (Offset = 38h) [Reset = 00000000h]

CLB_DBG_C0 is shown in Figure 26-63 and described in Table 26-66.

Return to the Summary Table.

Count of Unit 0

**Figure 26-63. CLB_DBG_C0 Register**

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

**Table 26-66. CLB_DBG_C0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | DBG | R | 0h | CLB_DBG_C0<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.3.20 CLB_DBG_C1 Register (Offset = 3Ah) [Reset = 00000000h]

CLB_DBG_C1 is shown in Figure 26-64 and described in Table 26-67.

Return to the Summary Table.

Count of Unit 1

**Figure 26-64. CLB_DBG_C1 Register**

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

**Table 26-67. CLB_DBG_C1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | DBG | R | 0h | CLB_DBG_C1<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.3.21 CLB_DBG_C2 Register (Offset = 3Ch) [Reset = 00000000h]

CLB_DBG_C2 is shown in Figure 26-65 and described in Table 26-68.

Return to the Summary Table.

Count of Unit 2

**Figure 26-65. CLB_DBG_C2 Register**

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

**Table 26-68. CLB_DBG_C2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | DBG | R | 0h | CLB_DBG_C2<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.3.22 CLB_DBG_OUT Register (Offset = 3Eh) [Reset = 00010100h]

CLB_DBG_OUT is shown in Figure 26-66 and described in Table 26-69.

Return to the Summary Table.

Outputs of various units in the Cell

**Figure 26-66. CLB_DBG_OUT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| OUT7 | OUT6 | OUT5 | OUT4 | OUT3 | OUT2 | OUT1 | OUT0 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| LUT42_OUT | FSM2_LUTOUT | FSM2_S1 | FSM2_S0 | COUNT2_MATCH1 | COUNT2_ZERO | COUNT2_MATCH2 | RESERVED |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-1-1h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| LUT41_OUT | FSM1_LUTOUT | FSM1_S1 | FSM1_S0 | COUNT1_MATCH1 | COUNT1_ZERO | COUNT1_MATCH2 | RESERVED |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-1-1h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| LUT40_OUT | FSM0_LUTOUT | FSM0_S1 | FSM0_S0 | COUNT0_MATCH1 | COUNT0_ZERO | COUNT0_MATCH2 | RESERVED |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0-0h |

**Table 26-69. CLB_DBG_OUT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | OUT7 | R | 0h | CELL Output 7<br>Reset type: SYSRSn |
| 30 | OUT6 | R | 0h | CELL Output 6<br>Reset type: SYSRSn |
| 29 | OUT5 | R | 0h | CELL Output 5<br>Reset type: SYSRSn |
| 28 | OUT4 | R | 0h | CELL Output 4<br>Reset type: SYSRSn |
| 27 | OUT3 | R | 0h | CELL Output 3<br>Reset type: SYSRSn |
| 26 | OUT2 | R | 0h | CELL Output 2<br>Reset type: SYSRSn |
| 25 | OUT1 | R | 0h | CELL Output 1<br>Reset type: SYSRSn |
| 24 | OUT0 | R | 0h | CELL Output 0<br>Reset type: SYSRSn |
| 23 | LUT42_OUT | R | 0h | LUT4_OUT UNIT 2<br>Reset type: SYSRSn |
| 22 | FSM2_LUTOUT | R | 0h | FSM_LUT_OUT UNIT 2<br>Reset type: SYSRSn |
| 21 | FSM2_S1 | R | 0h | FSM_S1 UNIT 2<br>Reset type: SYSRSn |
| 20 | FSM2_S0 | R | 0h | FSM_S0 UNIT 2<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

**Table 26-69. CLB_DBG_OUT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19 | COUNT2_MATCH1 | R | 0h | COUNT_MATCH1 UNIT 2<br>Reset type: SYSRSn |
| 18 | COUNT2_ZERO | R | 0h | COUNT_ZERO UNIT 2<br>Reset type: SYSRSn |
| 17 | COUNT2_MATCH2 | R | 0h | COUNT_MATCH2 UNIT 2<br>Reset type: SYSRSn |
| 16 | RESERVED | R-1 | 1h | Reserved |
| 15 | LUT41_OUT | R | 0h | LUT4_OUT UNIT 1<br>Reset type: SYSRSn |
| 14 | FSM1_LUTOUT | R | 0h | FSM_LUT_OUT UNIT 1<br>Reset type: SYSRSn |
| 13 | FSM1_S1 | R | 0h | FSM_S1 UNIT 1<br>Reset type: SYSRSn |
| 12 | FSM1_S0 | R | 0h | FSM_S0 UNIT 1<br>Reset type: SYSRSn |
| 11 | COUNT1_MATCH1 | R | 0h | COUNT_MATCH1 UNIT 1<br>Reset type: SYSRSn |
| 10 | COUNT1_ZERO | R | 0h | COUNT_ZERO UNIT 1<br>Reset type: SYSRSn |
| 9 | COUNT1_MATCH2 | R | 0h | COUNT_MATCH2 UNIT 1<br>Reset type: SYSRSn |
| 8 | RESERVED | R-1 | 1h | Reserved |
| 7 | LUT40_OUT | R | 0h | LUT4_OUT UNIT 0<br>Reset type: SYSRSn |
| 6 | FSM0_LUTOUT | R | 0h | FSM_LUT_OUT UNIT 0<br>Reset type: SYSRSn |
| 5 | FSM0_S1 | R | 0h | FSM_S1 UNIT 0<br>Reset type: SYSRSn |
| 4 | FSM0_S0 | R | 0h | FSM_S0 UNIT 0<br>Reset type: SYSRSn |
| 3 | COUNT0_MATCH1 | R | 0h | COUNT_MATCH1 UNIT 0<br>Reset type: SYSRSn |
| 2 | COUNT0_ZERO | R | 0h | COUNT_ZERO UNIT 0<br>Reset type: SYSRSn |
| 1 | COUNT0_MATCH2 | R | 0h | COUNT_MATCH2 UNIT 0<br>Reset type: SYSRSn |
| 0 | RESERVED | R-0 | 0h | Reserved |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.4 CLB_DATA_EXCHANGE_REGS Registers

Table 26-70 lists the memory-mapped registers for the CLB_DATA_EXCHANGE_REGS registers. All register offset addresses not listed in Table 26-70 should be considered as reserved locations and the register contents should not be modified.

**Table 26-70. CLB_DATA_EXCHANGE_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h + formula | CLB_PUSH_y | CLB_PUSH FIFO Registers (from HLC) | | Go |
| 100h + formula | CLB_PULL_y | CLB_PULL FIFO Registers (TO HLC) | | Go |

Complex bit access types are encoded to fit into small table cells. Table 26-71 shows the codes that are used for access types in this section.

**Table 26-71. CLB_DATA_EXCHANGE_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. |



Copyright © 2024 Texas Instruments Incorporated

### 26.8.4.1 CLB_PUSH_y Register (Offset = 0h + formula) [Reset = 00000000h]

CLB_PUSH_y is shown in Figure 26-67 and described in Table 26-72.

Return to the Summary Table.

CLB_PUSH FIFO Registers (from HLC)

Offset = 0h + (y * 2h); where y = 0h to 3h

**Figure 26-67. CLB_PUSH_y Register**

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

**Table 26-72. CLB_PUSH_y Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | PUSH | R | 0h | FIFO TO System From CLB<br>Note: The PUSH FIFO register does not get reset, so random values are expected upon power-on reset.<br>Reset type: SYSRSn |



Copyright © 2024 Texas Instruments Incorporated

#### 26.8.4.2 CLB_PULL_y Register (Offset = 100h + formula) [Reset = 00000000h]

CLB_PULL_y is shown in Figure 26-68 and described in Table 26-73.

Return to the Summary Table.

CLB_PULL FIFO Registers (TO HLC)

Offset = 100h + (y * 2h); where y = 0h to 3h

**Figure 26-68. CLB_PULL_y Register**

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

**Table 26-73. CLB_PULL_y Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | PULL | R/W | 0h | FIFO From system TO CLB<br>Note: The PULL FIFO register does not get reset, so random values are expected upon power-on reset.<br>Reset type: SYSRSn |

### *26.8.5 CLB Registers to Driverlib Functions*

**Table 26-74. CLB Registers to Driverlib Functions**

| File | Driverlib Function |
|---|---|
| **COUNT_RESET** | |
| clb.h | CLB_selectCounterInputs |
| **COUNT_MODE_1** | |
| clb.h | CLB_selectCounterInputs |
| **COUNT_MODE_0** | |
| clb.h | CLB_selectCounterInputs |
| **COUNT_EVENT** | |
| clb.h | CLB_selectCounterInputs |
| **FSM_EXTRA_IN0** | |
| clb.h | CLB_selectFSMInputs |
| **FSM_EXTERNAL_IN0** | |
| clb.h | CLB_selectFSMInputs |
| **FSM_EXTERNAL_IN1** | |
| clb.h | CLB_selectFSMInputs |
| **FSM_EXTRA_IN1** | |
| clb.h | CLB_selectFSMInputs |
| **LUT4_IN0** | |
| clb.h | CLB_selectLUT4Inputs |
| **LUT4_IN1** | |
| clb.h | CLB_selectLUT4Inputs |
| **LUT4_IN2** | |
| clb.h | CLB_selectLUT4Inputs |
| **LUT4_IN3** | |
| clb.h | CLB_selectLUT4Inputs |
| **FSM_LUT_FN1_0** | |
| clb.h | CLB_configFSMLUTFunction |



Copyright © 2024 Texas Instruments Incorporated

**Table 26-74. CLB Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| **FSM_LUT_FN2** | |
| clb.h | CLB_configFSMLUTFunction |
| **LUT4_FN1_0** | |
| clb.h | CLB_configLUT4Function |
| **LUT4_FN2** | |
| clb.h | CLB_configLUT4Function |
| **FSM_NEXT_STATE_0** | |
| clb.h | CLB_configFSMNextState |
| **FSM_NEXT_STATE_1** | |
| clb.h | CLB_configFSMNextState |
| **FSM_NEXT_STATE_2** | |
| clb.h | CLB_configFSMNextState |
| **MISC_CONTROL** | |
| clb.h | CLB_configMiscCtrlModes |
| **OUTPUT_LUT_0** | |
| clb.h | CLB_configOutputLUT |
| **OUTPUT_LUT_1** | |
| - | See OUTPUT_LUT_0 |
| **OUTPUT_LUT_2** | |
| - | See OUTPUT_LUT_0 |
| **OUTPUT_LUT_3** | |
| - | See OUTPUT_LUT_0 |
| **OUTPUT_LUT_4** | |
| - | See OUTPUT_LUT_0 |
| **OUTPUT_LUT_5** | |
| - | See OUTPUT_LUT_0 |
| **OUTPUT_LUT_6** | |
| - | See OUTPUT_LUT_0 |
| **OUTPUT_LUT_7** | |
| - | See OUTPUT_LUT_0 |
| **HLC_EVENT_SEL** | |
| clb.h | CLB_configHLCEventSelect |
| **LOAD_EN** | |
| clb.h | CLB_enableCLB |
| clb.h | CLB_disableCLB |
| clb.h | CLB_writeInterface |
| **LOAD_ADDR** | |
| clb.h | CLB_writeInterface |
| **LOAD_DATA** | |
| clb.h | CLB_writeInterface |
| **INPUT_FILTER** | |
| clb.h | CLB_selectInputFilter |
| clb.h | CLB_enableSynchronization |
| clb.h | CLB_disableSynchronization |
| **IN_MUX_SEL_0** | |



Copyright © 2024 Texas Instruments Incorporated

**Table 26-74. CLB Registers to Driverlib Functions (continued)**

| File | Driverlib Function |
|---|---|
| clb.h | CLB_configGPInputMux |
| **LCL_MUX_SEL_1** | |
| clb.h | CLB_configLocalInputMux |
| **LCL_MUX_SEL_2** | |
| clb.h | CLB_configLocalInputMux |
| **BUF_PTR** | |
| clb.c | CLB_clearFIFOs |
| **GP_REG** | |
| clb.h | CLB_setGPREG |
| clb.h | CLB_getGPREG |
| **OUT_EN** | |
| clb.h | CLB_setOutputMask |
| **GLBL_MUX_SEL_1** | |
| clb.h | CLB_configGlobalInputMux |
| **GLBL_MUX_SEL_2** | |
| clb.h | CLB_configGlobalInputMux |
| **INTR_TAG_REG** | |
| clb.h | CLB_getInterruptTag |
| clb.h | CLB_clearInterruptTag |
| **LOCK** | |
| clb.h | CLB_enableLock |
| **DBG_R0** | |
| - | |
| **DBG_R1** | |
| - | |
| **DBG_R2** | |
| - | |
| **DBG_R3** | |
| - | |
| **DBG_C0** | |
| - | |
| **DBG_C1** | |
| - | |
| **DBG_C2** | |
| - | |
| **DBG_OUT** | |
| clb.h | CLB_getOutputStatus |
| **PUSH** | |
| clb.c | CLB_readFIFOs |
| **PULL** | |
| clb.c | CLB_clearFIFOs |
| clb.c | CLB_writeFIFOs |



Copyright © 2024 Texas Instruments Incorporated

# Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

| Changes from October 25, 2023 to May 21, 2024 (from Revision J (October 2023) to Revision K (May 2024)) | Page |
|---|---|
| • Change OSCCLK cycles to INTOSC1 cycles in Section 3.9.3 | 120 |
| • Change OSCCLK cycles to INTOSC1 cycles in Section 3.9.4 | 120 |
| • Changed Figure 19-10 | 2313 |
| • Changed Figure 20-18 | 2359 |
| • Moved Figure 25-15 to Section 25.3.1 | 2836 |



Copyright © 2024 Texas Instruments Incorporated

# 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 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